(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 869 652 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
07.10.1998 Bulletin 1998/41

(51) Int. Cl.[6]: **H04L 29/06**, G06F 17/30, G06F 17/60

(21) Application number: **98105986.8**

(22) Date of filing: **01.04.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **01.04.1997 US 829976**
04.04.1997 US 832784

(71) Applicant:
Tumbleweed Software Corporation
Redwood City, California 94063 (US)

(72) Inventors:
- **Smith, Jeffrey C.**
  Menlo Park, Cal. 94025 (US)
- **Bandini, Jean-Christophe**
  Cupertino, Cal. 95014 (US)

(74) Representative:
DIEHL GLAESER HILTL & PARTNER
Flüggenstrasse 13
80639 München (DE)

(54) Document delivery system

(57) A document delivery system delivers an electronic document between a sending computer (16) and a receiving computer (22). A server (12) is interposed between the sending computer (I6) and the receiving computer (22). When an electronic document is forwarded to the server (12) from the sending computer (I6) the server (I2) dynamically generates a private Uniform Resource Locater ("PURL") to distribute the electronic document. The intended recipient (22) of the document uses the PURL to retrieve the document. The server (12), upon retrieval of the document, customizes the behavior of the retrieval based upon attributes included in the PURL. A method and system are provided for secure document delivery over a wide area network, such as the Internet.

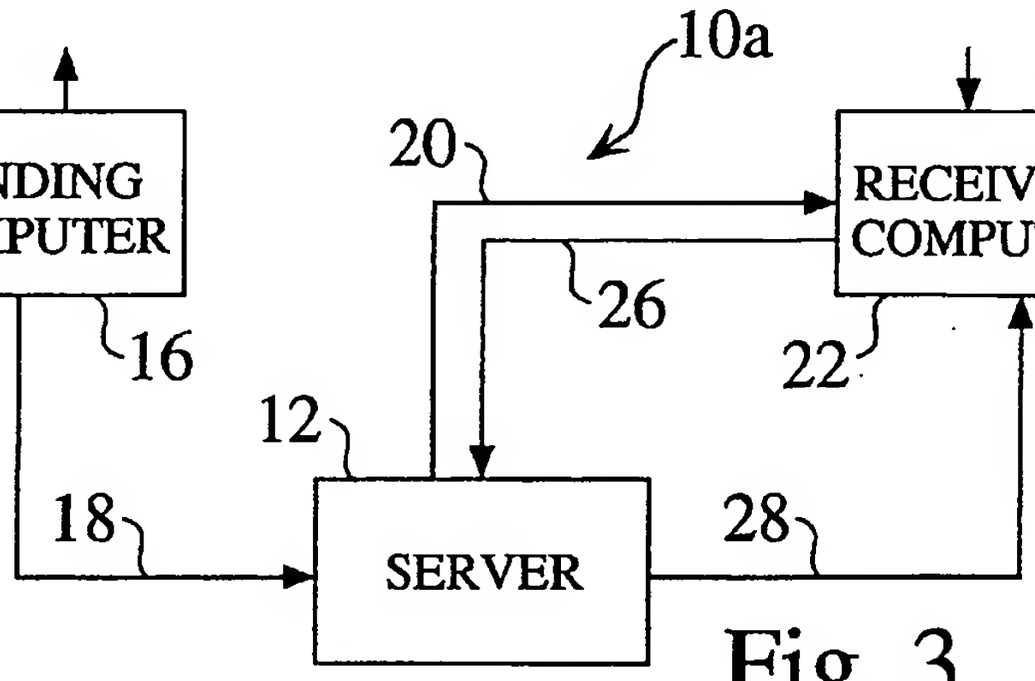

Fig. 3

EP 0 869 652 A2

Description

The invention relates to the field of computer networks.

The development of computerized information sources, such as those provided through the Internet or other on-line sources, has led to a proliferation of electronically available information. Currently, a user who subscribes to the Internet manually navigates through the Internet to visit sites which may or may not be of interest.

An inherent problem in this Internet system is that the available information is distributed through a "pull" type infrastructure, where the user who wants to receive information must manually search sites of interest, or use a finder application, to search and download appropriate information. For a user who wishes to publish and distribute information or documents, either an individual or a larger entity that has information that is desired to be distributed, the present "pull" system doesn't allow the freedom to send and distribute to a recipient or group of recipients, in a "push" fashion.

Facsimile technology is widely used at the present time for the distribution of simple documents, but has numerous drawbacks, including lower quality printed documents, costly and bulky paper copies (particularly if the recipient doesn't care to have a paper copy), loss of content (e.g. text and graphics can't be edited or manipulated), and time requirements for transmission, particularly for long or complex documents.

Electronic Mail (E-mail) provides a means for sending electronic messages from computer user to another. E-mail has advantages of convenience, format and storage of messages for later retrieval. As such, E-mail has been accepted and widely used for basic communication. E-mail is typically an ASCII based format, however, and proves to be very limiting for the communication of long or formatted documents. As well, E-mail is not the medium of choice for the distribution of complex documents, such as reports, articles, advertisements and art which can include page layout grids, postscript-formatted objects, multiple fonts with tracking and kerning, graphics, imbedded tables and spreadsheets, and other complicated information. Some E-mail systems provide a means for appending an ASCII based E-mail message with an associated file, to be downloaded along with the E-mail message. Most systems that allow the appending of an associated file are designed to allow a single user to send unsecured files to an associate or friend, and neither allow for controlled automated distribution to multiple recipients, nor do they provide advanced accounting, billing or other such features (e.g., receipt notification). E-mail gateways also limit the applicability of attachments, and do not solve the problems of security and receipt notation or acknowledgment.

C. Baudoin, *Interenterprise Electronic Mail Hub*, U.S. Patent No. 5,406,557 (11 April 1995) discloses an interenterprise communications center, which has a computer hub comprising a common core and a plurality of input and output modules. The input modules connect to a first end user, and convert a message sent by the first end user into a universal format. The hub core queues the message and forwards it to the output module for conversion into the format of the destination user. While the disclosed hub discloses techniques to relay simple e-mail messages, it is designed to convert the e-mail message formats, thus losing the integrity of the original textbased file.

The disclosed prior art systems and methodologies thus provide some methods for the delivery of documents, but fail to provide an economical, fast document delivery system that operates in a push-fashion, while conserving the integrity of the original electronic file. The development of such an electronic document delivery system would constitute a major technological advance. In addition, the ability to distribute electronic portable high content- quality documents to many recipients in a controlled, economical and accountable fashion would constitute a further technological advance.

The Internet is increasingly being used for communications. It is now possible on the Internet for a sender to direct a document to a specific recipient, regardless of platform, operating system, or email system. The sender's computer may be connected to the Internet directly, or through an intranet's server. Such communication is possible even when the recipient is not a computer but, rather, a fax machine or printer connected to the Internet.

This increase in Internet communications has necessitated the development of security systems to insure protection for information transmitted over the Internet.

Encryption is a basic technique used to scramble information to prevent unsolicited access to that information. One well-known encryption scheme is secret key encryption, sometimes referred to as private key encryption or symmetric-key cryptography. Secret key encryption employs the technique of scrambling information using a unique key to prevent unsolicited access thereto. This unique key is then required to unscramble the information. Figure 1 is a diagram illustrating secret key encryption, according to the prior art.

A document 1010 is scrambled 1012 using a secret key 1014. A secret key is an encryption scheme that is only available to authorized users of the scheme. The encryption software may be located on the user's computer, or at a remote location. Thus, the document may be encrypted in situ, or upon transmission to another computer, such as an intranet server.

The resulting encrypted document 1016 is then transmitted to the recipient. It is unscrambled 1018 using the secret key 1014 to regenerate the original document 1010. The encrypted document cannot be accessed without the secret key. Again, the decryption software may be located on the recipient's computer, or at a remote location.

One potential problem associated with secret key encryption is the secure distribution of the secret key. If the secret

key is sent over a non-secure channel, the integrity of the security is compromised. For most practical applications, telephone or fax provides adequate security for delivering secret keys, while the document can be delivered over the internet using such mail schemes as Posta, which is available from Tumbleweed Software Corporation of Redwood City, CA. In some instances, however, users require a more secure, or more convenient, means of distributing a key.

Another known encryption scheme is public key encryption. In public key encryption, the sender and the recipient each own a pair of keys, called the public key and the private key. The owner of a key pair publishes the public key and keeps the private key a secret.

The sender uses the published public key of the intended recipient to encrypt information. The information is decrypted using the recipient's private key. Thus, using public key encryption, no private key must be distributed.

Figure 2 is a diagram illustrating public key encryption, according to the prior art. A document 1020 is scrambled 1022 using a public key 1024. The resulting encrypted document 1026 is then transmitted to the recipient. It is unscrambled 28 using the private key 1030 to regenerate the original document 1020.

The keys used in public key encryption are very large numbers. Public key encryption exploits an esoteric mathematical relationship between the key numbers to implement the encryption and decryption. As a result, the private key cannot readily be derived from the published public key.

It is often useful to verify that a document has not been altered during transmission, or to verify the sender or recipient of a document. Secret and public key technology provide such verification. However, public key encryption algorithms are typically complex and often are too time consuming to be of practical use. Secret key encryption is much faster, but there are difficulties associated with securely transmitting the key.

A public key/private key encryption system is described in *Ganesan*, Yaksha, An Improved System And Method For Securing Communications Using Split Private Key Asymmetric Cryptography, U.S. Patent No. 5,535,276 (9 July 1996). However, the *Ganesan* encryption scheme uses a complicated scheme for generating temporary keys and requires several different users to manually request public keys.

In *Torii* , Key Distribution Protocol For File Transfer In The Local Area Network, U.S. Patent No. 5,313,521 (17 May 1991) a key distribution center is used to authenticate a terminal to a server. *Pastor,* Reliable Document Authentication System, U.S. Patent No. 4,853,961 (1 August 1989) describes a document authentication system that includes a decryption key. Choudhury, et al, Method of Protecting Electronically Published Materials Using Cryptographic Protocols, U.S. Patent No. 5,509,074 (16 April 1996) teaches a document protection system that includes a server-to-server security access operation to authenticate each document request. However, all of these prior art schemes require user intervention to authenticate the certificate.

Another encryption scheme, digital envelopes, is not subject to the disadvantages of secret key and public key encryption. Using digital envelopes, a sender encrypts a document with a secret key. The secret key is then encrypted with a public key. The recipient of the document then uses the recipient's private key to decrypt the secret key, and then the secret key to decrypt the document.

Registries are now available for publication of public keys. Such registries can certify that a particular public key belongs to a particular entity. For example, a certificate authority issues and maintains digital certificate that are used to connect entities to their specific public keys. The sender must query the registry to receive the requested public key information. This time-consuming process is inefficient, especially when the sender has a large number of documents to transmit to different recipients.

It is therefore the object of the present invention to overcome the drawbacks and disadvantages of the prior art. This object is solved by an apparatus for delivering an electronic document according to independent claim l, a document delivery system and a method for delivering an electronic document according to independent claims l2 and 30, methods and systems for secure document delivery according to independent claims 37, 48, 54 and 63 respectively.

Further advantageous features, aspects and details of the invention are evident from the dependent claims, description and drawings. The claims are to be understood as a first non-limiting approach of defining the invention in general terms. The invention relates to techniques for the delivery of electronic documents to users over the Internet. Further, the invention relates to a method and system for providing secure document delivery over a wide area network, such as the Internet.

It is an advantage to provide a system and method for automatically and dynamically retrieving a public key over a wide area network for encryption purposes. It is a further advantage if such system and method uses a server to retrieve the certificate and requires no user intervention. It is yet another advantage if the system and method does not transmit a document to the server until the server has returned the public key to the user.

An electronic document delivery system and methods of its use are provided. A document delivery architecture dynamically generates a private Uniform Resource Locator (URL) to distribute information. Each private URL ("PURL") uniquely identifies an intended recipient of a document, the document or set of documents to be delivered, and optionally other parameters specific to the delivery process. The intended recipient of a document uses the PURL to retrieve the document. The server, upon retrieval of the document, customizes the behavior of the retrieval based upon attributes included in the PURL, as well as log information associated with the retrieval in a data base. This architecture

and usage of PURLs enables secure document delivery and tracking of document receipt.

The invention provides a method and system for secure document delivery over a wide area network, such as the Internet. A document is sent from sender to recipient via a Delivery Server. In the preferred embodiment of the invention, the Delivery Server is directed by the sender to retrieve the intended recipient's public key (certificate). The Delivery Server dynamically queries a certificate authority and retrieves the public key. The public key is transmitted from the Delivery Server to the sender.

The sender encrypts the document using a secret key. The secret key is then encrypted using the public key. Both encrypted document and encrypted secret key are uploaded to the Delivery Server, and transmitted to the intended recipient. The intended recipient then uses the private key associated with the public key to decrypt the secret key, and uses the secret key to decrypt the document.

In an alternative, equally preferred embodiment of the invention, the sender uses the public key to encrypt the document. The encrypted document is then transmitted to the intended recipient and decrypted using the private key associated with the public key.

In yet another embodiment, the server transmits the document to the Delivery Server for encryption. The Delivery Server queries the certificate authority in real time to retrieve the public key. The Delivery Server encrypts the document using a secret key, and then uses the public key to encrypt the secret key. The Delivery Server then transmits the encrypted document and the encrypted secret key to the intended recipient.

In the event that the Delivery Server query returns failure (no certificate available for the given user), the Delivery Server dynamically generates a new public key for the intended recipient. This new certificate is then used to encrypt the document.

Figure 1 is a diagram illustrating secret key encryption according to the prior art;

Figure 2 is a diagram illustrating public key encryption according to the prior art;

Figure 3 is a block diagram which depicts a binary file delivery system using one binary file server;

Figure 4 is a block diagram which depicts a binary file delivery system using two binary file servers;

Figure 5 is a block diagram which illustrates key elements of a store item;

Figure 6 is a schematic depiction of the binary file delivery server;

Figure 7 provides an example of the architecture of one embodiment of the binary file server;

Figure 8 illustrates different types of store events employed by the binary file delivery server;

Figure 9 is a block diagram of the specific components within the binary file delivery server architecture;

Figure 10 provides a block diagram illustrating of the architecture of the store;

Figure 11 illustrates how the user session organizes internet clients into three layers, including sessions, transactions, and transports;

Figure 12 illustrates the non-interactive tasks of a delivery, once the send session has created a store item or another server is forwarding a store item;

Figure 13 provides details of the account manager architecture;

Figure 14 provides details of the logger architecture;

Figure 15 provides details of the server connector architecture;

Figure 16 provides a functional block diagram which depicts a portable document delivery system using one portable document delivery server;

Figure 17 provides a functional block diagram which depicts a portable document delivery system using two portable document delivery servers;

Figure 18 illustrates how a portable document send client application and a portable document receive client application are used in the invention;

Figure 19 illustrates how a server configuration user interface application is used in the invention;

Figure 20 illustrates how a document can be sent by the fax gateway of a server to a printer;

Figure 21 illustrates how a document can be sent by the department gateway of a dedicated corporate server through a LAN to a department printer;

Figure 22 is a block diagram which depicts a document delivery system that includes private, trackable URLs for directed document delivery according to the invention;

Figure 23 is a diagram illustrating dynamic server document encryption according to a first preferred embodiment of the invention;

Figure 24 is a flow chart of the set of operations for dynamic server document encryption according to a first preferred embodiment of the invention; and

Figure 25 is a flow chart of the set of operations for dynamic server document encryption according to an alternative embodiment of the invention.

The binary file delivery system 10 enables corporations, publishers and individuals to distribute documents electronically. Importantly, unlike existing Web based document publishing technologies, the binary file delivery system 10 allows the directed and secure distribution of documents. The Web could currently be characterized as a pull-publishing environment, where the consumer of documents must find and retrieve documents from a server. Push-publishing, by contrast, allows the producer of a document to direct the delivery of documents to consumers. Facsimile (fax), the postal service, and electronic mail (E-mail) are all examples of push-publishing.

Figure 3 is a block diagram which depicts a binary file delivery system 10 using one binary file server 12. The binary file delivery system 10 allows users to push documents, enabling the producer of documents to direct where those documents will go. One way that the binary file delivery system 10 achieves push-publishing is by combining HTTP, which is usually implemented to pull information over a network, with SMTP (which only supports text). Additionally, the binary file delivery system 10 provides a host of services to facilitate the various applications of directed document delivery. At one level, the binary file delivery system 10 can be characterized as a new generation of facsimile technology, which utilizes networks instead of telephone lines, and moreover, introduces support for new document representations vastly superior to existing fax formats. At another level, the binary file delivery system 10 is a general purpose document delivery server capable of supporting massive amounts of documents and transactions. In all cases, the binary file delivery system 10 provides a complete and robust solution for document delivery.

The binary file delivery system 10 is used for sending a set of binary files from one end-point to one or multiple endpoints. An endpoint is typically a recipient 22 with Internet access, but can also be another entity, such as a facsimile machine 172 or a printer 178 (Figs. 16, 17). The delivery of binary files is accomplished in a reliable, accountable, and tractable manner. The binary file delivery system 10 provides several levels of security for the directed files, from E-mail equivalent security, to better than facsimile or physical mail. The system also provides user account management including the credit and debit of billing accounts. The system can also cooperate between multiple binary file delivery servers 12, which may or may not be controlled by some other authority. Figure 4 depicts a binary file delivery system using two binary file servers 12a and 12n, which communicate across an Internet.

The binary file delivery server 12 operates in three primary modes, which include a public mode, where senders 16 set up their accounts 132 themselves and are subject to billing, a private mode, where senders 16 are controlled by an administrator, and billing is more an internal accounting issue than a collection issue, and a publishing mode, where there are many recipients 22, but few senders 16.

The binary file delivery server 12 is comprised of separate functional components, and are not necessarily processes or shared libraries. The binary file delivery server 12, shown schematically in Figure 6, includes an intelligent storage compartment called a store 42, which is augmented by a set of clients 44a-44n, called store clients 44, which use the store methods and listen to the store events, but do not interact with or know about other clients 44. An account manager 46 component is a shared service that keeps information about the sender 16. The design also incorporates information about recipients 22 for the case of a receive application (as opposed to e-mail notification).

The client/server general architecture provides a better extensibility than a more pipelined structure. It also decouples the store clients 44 from each other, which can be useful in the context where some tasks are interactive, while

others are more background oriented.

The Store. The store 42 contains a set of store items 48. As shown in Figure 5, a store item 48 includes a tree of binary files 34 and a descriptor 36, which is a set of store-defined and client-defined attributes. The tree of binary files 34 can be viewed as part of the store-defined attributes.

The file storage system provides the following functionality:

1) Permanent storage of Store items 48 (*e.g.* the binary file tree 34 contained in a store item 48 is written to disk)

2) Client read/write access to the descriptor 36, which is made up of store-defined and client-defined attributes (*e.g.* a client 44 can write the expiration date of a store item 48)

3) Client notification of store events 67 (*e.g.* clients 44 can be notified of the creation event 68 of a new store item 48)

4) Internal management according to store defined attributes (*e.g.* store item expiration date generates an event).

The store 42 provides access to the store items 48 and generates store events 67, wherein store items 48 have store-defined attributes such as ID, creation date, file count, file names, file data, and store events 67 can be listened to by the clients 44. Store events 67 may include the creation 68, deletion 69 or modification 70 of a store item 48. The events 67 play a crucial role in the architecture, since this defines how the clients 44 synchronize their work with a very limited knowledge of the other.

Store Clients. Store clients 44 can be of a wide variety, and specific clients will be detailed further. In this framework, a store client 44 is some component which uses some of the store methods and or listens to some of the store events 67 to perform useful tasks on the store items 48.

Account Manager. The account manager 46 provides read/write access to user and billing accounts, and is used by clients 44 or other components of the system 10. The store 42 does not use or know about the accounts.

Other Components. Other components used by the store clients 44 and the store 42 itself are implemented within the architecture of the system. For example, interserver communication, log management, and other administrative services, which is discussed below.

Figure 7 provides an example of the architecture of one embodiment of the binary file server 42, including client 44 modules (52-66) that are used to implement server functions. The Internet Send 52 is used to create store items 48 and fills in the attributes. The Internet Receive 54 opens existing store items 48 and can be used to modify their attributes. A Fax gateway 56 listens to the creation events 68 generated by the store 42, processes relevant store items 48, and then deletes them from the store 42. A forwarder 58 listens to the creation events 68 generated by the store 42, and then examines the attributes of the new store items 48, and decides if forwarding is necessary. An archiver 60 listens to deletion events, and copies the store item 48 to secondary private storage before deletion occurs. The format translator 62 listens to creation, examines attributes, and if translation is needed, it reads, processes and writes back the files in the store item 48. The web publisher 64 listens to the creation events 68 and checks if the store item attributes specified a Web publishing, and if so, read the attributes as necessary. A pickup notifier 66 listens for a creation event 68, and then notifies recipients 22.

Security Issues for Internet-based Users. While the Binary File Delivery System 10 offers the flexibility to support specialized security solutions, it readily supports current industry-standard security solutions, including:

a) secure server interconnect and server authentication (available with SSL 2.0, which is built into the servers (HTTP);

b) secure Server-to-Server (on top of SSLX);

c) support end-points private key (the private key has to be exchanged by the users using their own channels;

d) support end-points public key, using CryptoAPI or the standard user public key. The system can also help the user generate a public key for BFD-use only, and update user account information with it, so that the sender does not have to communicate directly with the recipient to get the public key; and

e) Client Authentication by the server with SSL and MS PCT (End user can get their own certificate and be authenticated by the servers).

An important aspect of the binary file delivery server 12 is that it handles multiple requests in parallel and minimizes

the response time for most requests. Therefore, synchronization issues are important, for both correctness and system performance. Performance is enhanced by minimizing synchronized data access, deferring to asynchronous processing whenever possible, and by using multitasking and Inter-Process Communication (IPC) for the platform. One embodiment of the server 12 relies heavily on threading, which provides low overhead multitasking within one process, and leverages multiprocessor capabilities when available. The IPC on this embodiment uses named pipes, in addition to mail slots or Remote Procedure Call (RPC).

Figure 7 provides a block diagram of the specific components within the binary file delivery server 12 architecture.

The user session 72 handles send sessions, receive sessions (which are implemented when the user is using a BFD desktop application 192, 198), HTML receive sessions (which are implemented through an HTML browser, as opposed to when the user is using the BFD desktop 164 (note that a BFD desktop session may go through HTML)), maintenance sessions (which implement the account setup and maintenance sessions (*e.g.* notification downloads, account setting modifications (not to be confused with console services by an administrator, as opposed to an end user of a public server), HTML maintenance sessions (which implement the account setup and maintenance through an HTML browser).

A delivery component 74 implements the background work of making a delivery, including notification and forwarding. A console 76 is used to implement administration sessions, which are conducted through an HTML interface instead of a specialized user interface. The console 76 provides a user interface to browse and modify all the server properties, including accounts, logging, performance, and parameter settings.

Shared Components. Shared components may be used by the store 42, by any of the store clients 44, or they may operate on their own. While they do not listen to store events 67, they can use store methods, as needed, for efficiency, such as for connector receive). Shared components may include:

1) An account manager which maintains all local account information and provides a unique access interface to local accounts, including billing account and remote account information;

2) a server connector 80, which handles all inter-server communications;

3) a mail gateway 84, which handles the sending and receiving of bounced mail;

4) a logger 86, which manages access read/write to the different logs which are classified by a type. The most important log is the send/receive transaction log, which tracks what happens to store items 48 over time; and

5) an operating system accessor 82, which provides a platform independent interface to the operating system for file input and output (I/O), process management (synchronization, locking, threads, process), PC (RPC, shared memory, shared queues, pipes), network access (TCP/IP sockets, HTTP server interfacing, POP/SMTP interfacing). Specific portions will be implemented as needed.

The Server Application. The server application 88 is used to start up and shut down all pieces of the binary file delivery server 12, according to the configuration parameters. It also provides the administrative aspects of the server not covered by the Account Manager (46 or 78) or by the Logger 86, such as performance profiling, usage information and server parameters/configuration.

Figure 10 provides a block diagram illustrating of the architecture of the store 42. A store manager 92 is used to maintain global state, to synchronize access to the store 42 and to provide housekeeping functions. A store item manager 94 is used to maintain the state, locks, and cache mechanism of a store item 48. A store event manager 96 is used to maintain listener lists and event filters, as well as to dispatch events according to event filters and event priorities.

Figure 11 illustrates how the user session organizes internet clients into three layers, including sessions, transactions, and transports. The session manager 102 maintains all the currently active session states and performs the session-related housekeeping. It processes transactions coming from transaction managers 108 through the uses of the store 42 and the account manager 46. The transaction manager 108 receives raw data from the transport managers 114, 118, and performs validation and preprocessing using one or more BFD transaction interpreters 110 or HTML transaction interpreters 112. The transaction manager 108 then submits the data to the appropriate BFD session manager 104 or HTML session manager 106, waits for an answer, and then passes the answer back to the appropriate transport manager 114 or 118.

Figure 12 illustrates the non-interactive tasks 120 of a delivery, once the send session has created a store item 48 or another server 12 a-n is forwarding a store item 48. The delivery manager 122 listens to relevant store events, makes a forwarding decision, and coordinates work with the notifier 66 and the forwarder 58. The server directory keeps track of the association between E-mail domains and server domains. The notifier 66 is used to handle E-mail notification 20 to the recipient 22. The forwarder 58 is used to forward store items 48 to other servers 12a-n, using a server connector

80. Since not all E-mail notifications may be received, an E-mail scanner is used to check the server mail account for "returned" E-mail, and then to match it with the failed transaction.

Figure 13 provides details of the account manager architecture 130. The account manager 78 is used to maintain user account states 132 for the local server 12, to maintain billing account states 134 for the local accounts 132, to query local accounts 132, and to maintain a directory of remote accounts 136. The primary goal of the remote account directory 136 is to associate E-mail addresses with either BFD accounts or non-BFD accounts.

Figure 14 provides details of the logger architecture. Figure 15 provides details of the server connector architecture.

System Operation. The following example illustrates how the binary file delivery system 10 is used to distribute electronic information from a sender 16 to a receiver 22. A hypothetical publisher, Sam in Redwood City, California, wishes to send a document to an associate, Rob, in Tokyo, Japan. The following progression of events illustrates how this is achieved, in a controlled fashion.

Sam connects to a local server in Santa Clara, California. Sam's BFD desktop opens a connection to a local server 12a in Santa Clara, where his user account resides. The session manager 102 queries the account manager 78 to validate the user 16 (Sam). The session manager 102 then creates a send session state for the user 16.

Sam's Send Session. Sam's BFD desktop sends transaction details, such as the number of files, file size, and intended recipients. The session manager 102 attaches this data to the session state. Then the session manager 102 creates a store item 48 descriptor 36 in memory, and reserves disk space with the store 42, as well as a store item ID. Then the upload starts. The session manager 102 spools the data directly to a file with asynchronous I/O.

When the upload 18 of all of Sam's files is complete, the session manager 102 updates the store item descriptor 36 to the disk asynchronously, and then inserts the store item 48 asynchronously into the store 42.

The session manager 102 answer's Sam's upload with an acknowledgement, and provides information regarding the transaction. This session then ends.

At the Santa Clara Store. The insertion of the store item 48 is logged asynchronously in the logger 86 by the store 42. The store 42 then runs the store item descriptor 36 against the registered event handlers filters. For each match, it inserts the event and notifiee (Rob) in its event queue. Then that thread dies.

The event dispatch thread pulls the events, and dispatches them asynchronously to the notifiee at rate, depending on the tuning parameters of the system.

The Santa Clara Delivery is Notified. The delivery 74 is notified of a relevant event and starts a thread which waits on the lock of the store item 48 via a synchronous transaction with the store 42. Once the lock is secured, the thread reads the store item descriptor 36, and the delivery manager 74 analyzes it, to decide how to handle it. It turns out the recipient 22 is in the Japan domain, where another BFD server 12n is located. The delivery manager 74 found this out by querying a server directory 124. The manager then decides to forward the store item 48. The forward manager 80 asynchronously asks the Connector 80 to do a forward to Tokyo. Then the thread in the delivery dies. Note that the delivery does not know about the server protocols.

The Santa Clara Connector 80 is going to forward the Tokyo Connector 80. A thread handling the delivery request is eventually started in the Connector 80. It knows the host, and has a lock on the store item 48. It initiates the connection with the Tokyo server 12n. If it cannot connect, it goes to sleep for a while. Eventually, the connection opens, and the connector 80 enters the protocol interpreter, which eventually transfers the store item descriptor 36 and the associated binary data files 34. Then it closes the connection and logs a successful forward to the Tokyo server 12n in the logger 86. Then the connector 80 releases the lock on the store item 48 in the store 42 after having marked it as forwarded.

On release of the lock, the store 42 runs the store item descriptor 36 against the event filter list and finds an event filter that is handled locally. A successfully forwarded store item 48 causes a reference count decreased by 1. In this example, there is only one recipient 22, which means the count goes to zero. Therefore, the store 42 can move the store item 48 to a deletion list. A housekeeping thread of the store 42 will then purge the Store Item 48 at some point.

A thread in the Tokyo connector receiver 80 is begun, to handle the connection. Once the protocol interpreter understands it as a forward, it asks the store 42 for a store item ID 36 and the respective committed storage space. The actual store item descriptor 36 and files have been written to disk as it was receiving the data.

Once the connection is complete, the store item 48 is inserted asynchronously into the store 42 of the Tokyo binary file delivery server 12n.

Tokyo Delivery Component begins. The Tokyo store 42, on insertion, has generated an event which is going to be handled by a thread of the delivery. It has also logged the insertion of the new item in the logger 86. The manager 102 in delivery 74 realizes this has been forwarded, and that it will be received from this server 12n.

The server 12n queries the account manager 78 to see if there is an account associated with the E-mail address of Rob. If there is no associated account with Rob E-mail, then an E-mail is sent to Rob, with an URL which indicates the store item ID 36. It also queues an asynchronous request for the connector 80 to notify the Santa Clara server 12a that Rob has been notified. If Rob has an account here, then the delivery puts an asynchronous update request with

the account manager 78 to mention the pending delivery; in this case the scenario is continued.

Rob connects to the Tokyo Server to check on new documents. When Rob opens its receive session, the session manager 102 synchronously checks the Rob account for validity, and in the process it updates the session state, to remember that the account is flagged with a pending receive. The BFD desktop of Rob eventually asks for the document to be received. The session state has the answer and says yes.

The Rob desktop 170 asks for the receive, and the session manager 102 synchronously asks the store 42 for the lock on the relevant store item 48. Once granted, it can answer by sending the first portion of data. Once the document is downloaded, it asynchronously logs a successful receive with the logger 86. Then it puts an asynchronous request with the connector 80 to notify the Santa Clara server 12a of the final delivery.

At the receive session in Tokyo, the session manager 102 releases the lock, and puts an asynchronous delete request to the store 42. The Rob receive session then terminates. The connector 80 in Santa Clara runs the protocol interpreter, which says that the notifications must be queued to the logger 86.

Sam checks on Status. Sam connects to do a receive session followed by a maintenance session. The maintenance session 72 receives a request to check on the status of the sent document. The maintenance session 72 synchronously submits a query to the logger 86 using the store item ID 36 that was passed down to the Sam desktop at send time. The query returns the lists of matching records, which are processed and passed back to the desktop, which can then update the user interface 16.

Portable Document Delivery System. Electronic portable documents are becoming increasingly popular. These files can be distributed to different platforms without losing their original look and feel. Adobe Systems' Acrobat PDF™ and Novel's Envoy™ portable document formats have come into widespread use. In a preferred embodiment of the invention, a portable document delivery system 160 achieves a universal solution to the delivery of electronic documents, by applying portable document technology to the Internet. The portable document delivery system 160 provides complete compatibility with portable electronic document formats, including Novell's ENVOY™ and Adobe System's PDF™ formats.

Recipients 22 of portable documents from the portable document delivery system 160 can view, search, print, archive, or export information from their documents. Documents distributed using Envoy™ or Acrobat™ in conjunction with the portable document delivery system 160, preserve complete visual fidelity and may be produced on high resolution output devices with the highest level of quality and resolution. Portable document formats allow preserve content and color of the information within a document, and many formats allow indexing, searching, and hypertext linking, while allowing the file to be stored in a compact manner.

Figure 14 is a functional block diagram which depicts a portable document delivery system 160a using a binary file delivery server 12. Figure 15 provides a functional block diagram depicting a portable document delivery system 160b using two binary file delivery servers 12a and 12n communicating over the Internet.

To address the limitations of the Web and electronic mail, in addition to providing additional services, the portable document delivery system 160 includes server software which runs on top of existing electronic mail, http server software, and database systems. Thus, the portable document delivery system 160 combines industry standard solutions for the electronic mail, Web, and database to enable corporations and users to direct the delivery of documents to recipients.

The following disclosure elaborates on the requirements for a universal document delivery solution, as well as the specific components of the portable document delivery system 160.

The portable document delivery system 160 combines three basic components to provide a solution to universal document delivery.

1) **Portable Document Send Client**. A portable document send client (PDSC) 192 integrates all desktop applications 190 directly with the portable document delivery system 160. The PDSC 192 is not required for all embodiments of the invention. Publishers who simply wish to leverage the BFD server 12 directly are free to do so. The PDSC 192 is intended for the standard corporate computer user who requires a point-to-point to the delivery problem.

2) **Binary File Server**. The binary file delivery server 12 works on top of Internet standards to deliver documents to recipients. The BFD server 12 can be invoked transparently through the portable document send client (PDSC) 192, or can be invoked and customized directly using a server configuration user interface 198.

3) **Portable Document Receive Client**. The portable document receive client (PDRC) 194 is the software component which recipients 22 of documents utilize to receive, view, and print documents. Recipients 22 who do not have the PDRC software 194 will be given links to access the software directly over the Internet. In most cases, the PDRC 194 will behave simply as a Netscape NAVIGATOR™ Plug-in or a Microsott ActiveX™ control or a Java Applet, thus directly integrating the PDRC 194 with the recipient's existing browsers.

Figure 18 illustrates how a portable document send client application and a portable document receive client application are used in the invention. Figure 19 illustrates how a server configuration user interface application is used in the invention.

Portable Document Delivery System Requirements. At the most basic level, a document delivery solution must enable documents to be directed to customers by the producers of those documents, or "pushed". The portable document delivery system 160 is designed so that different types of recipients operating on different computer systems, with different operating systems, E-mail systems, and document types can all benefit from receiving, reading, and using electronic portable documents. The various design parameter categories that the portable document delivery system 160 is adapted for includes primary computer systems (*e.g.* PCs, Workstations, Servers), primary operating systems (*e.g.* Macintosh, Win 3.1, Win '95, NT, Unix, OS/2), electronic mail systems (*e.g.* Microsoft, cc:Mail, Groupwise, Motes, Eudora), document types (*e.g.* paper, Postscript, Quark, WordPerfect, Excel), and user types (*e.g.* MIS, Legal, Financial, Consumers/Home, MarketingCommunication (MarCom)).

A unique aspect of the portable document delivery system 160 is the level of compatibility the solution provides with all computer systems, operating systems, electronic mail systems, and document types. In one embodiment of the invention, the sender 16 and the receiver 22 of a document are both connected to the Internet. In a preferred embodiment of the invention, the portable document delivery system 160 provides not only an Internet delivery solution, but also backward compatibility with facsimile machines 172 and printers 178, as well as forward compatibility with future distribution print architectures.

Universal Delivery. Delivery solutions must enable users to distribute documents to anyone, which requires support for a variety of computing platforms, compatibility with facsimile 172, and compatibility with future distributed printing architectures. The portable document delivery system 160 can support the conversion and delivery of complex postscript files. Documents can be delivered to any recipient 22 who has an E-mail account and access to the Internet, regardless of the recipient's platform or E-mail system.

Security. Typical applications of document delivery require complete security from the origin of the document complete to the destination. This requirement becomes more pervasive as documents begin to travel across open and wide area networks. The portable document delivery system 160 employs several levels of security. The Portable Document Send Client 192 authenticates and creates a secure socket to upload information to the server 12. Thus, non-BFD servers cannot intercept documents. Additionally, The PDSC 192 allows the sender 16 to use private and or public encryption to guarantee that only the intended recipients of a document can access those documents. Even in cases where encryption is not used, the portable document delivery system 160 includes sophisticated algorithms to prevent unauthorized users from accessing documents.

Account Management Services. In many instances, document delivery applications cater to businesses where each sender 16 or recipient 22 of a document must be maintained. Consider the case of periodically delivering the documents to the same group of a hundred thousand recipients 22. The sender 16 of the document requires tools to update and manipulate the database of the large subscription/ distribution base.

The portable document delivery system 160 enables publishers 16 to create accounts on BFD servers 12 and then associate transactions with specific accounts 132, 134, 136. The system also enables publishers to consolidate several user accounts into a single billing account 134. Additionally, it allows publishers to associate a specific billing code with transactions which may be consolidated in transaction reports. For example, a law firm could create an account and then a billing code for each client, associating a billing code and account with each document's transaction. The portable document delivery system 160 maintains and updates the account information automatically. A portable document delivery system 160 reporting engine then allows the user to create a report for a given account or for a specific billing code. Such a scheme facilitates client management as well as billing.

Transaction Management Services. Related to account management is the requirement of transaction management. Not only is it necessary to maintain the database of senders 16 and recipients 22 of documents, it is also necessary to provide services to manage the transaction of sending documents. For example, a sender 16 may want to know if the document was actually delivered and actually received, and perhaps who received the document. In many instances, the publisher 16 would like to charge postage for delivery and will therefore require services to maintain and update accounting information associated with the delivery transactions.

The portable document delivery system 160 is able to create logs associated with each send transaction, and maintain these logs. Each transaction, or document send operation is associated with a specific account. Users 16 can query transaction information directly from the server.

Reporting. Account and transaction management provides no value unless sophisticated means of reporting are provided. For example, users 16 can be provided with a full report of a given transaction, including such information as which documents were delivered to whom, how many users have confirmed delivery of the document, or for billing purposes, the costs associated with the transaction.

Scalability and Bandwidth. Because of the large scope and application of document delivery applications, the portable document delivery system 160 is capable of expanding its capabilities to service millions of documents or

recipients 22. Several aspects of the delivery process occur in real time, while other aspects may be deferred or scheduled. In many cases, the portable document delivery system 160 dynamically extends the amount of bandwidth or sets of servers 12a-n deployed to achieve the necessary throughput for document delivery.

The portable document delivery system 160 is scalable to conform with user requirements. The server software is designed to support the sending of millions of documents per day, and is able to exploit whatever bandwidth has been dedicated to a given server. For example, one current BFD server 12 effectively utilizes 10 Megabit/second bandwidth. The various processes running on BFD servers 12 operate asynchronously, thus allowing for optimal performance on multi-processing servers 12, as well as sophisticated scheduling of the servicing of a given transaction. Special care is taken to operate in real time, particularly for the access of documents from the server 12 by recipients 22.

BFD servers 12 can also distribute work loads across other servers 12a-n. A preferred embodiment of the invention allows individual processes running on a single server 12 to be distributed across a collection of servers 12a-n. In this embodiment, account management processes could run on one server (*e.g.* 12d), while the logging, reporting, transaction management, send, propagate, and retrieve processes run on another server (*e.g.* 12h).

Portable Document Send Client Specification. The Portable Document Send Client (PDSC) 192 allows any computer user to distribute documents directly from the desktop of any personal computer, such as a PC or Macintosh computer. The PDSC 192 integrates directly with all applications 190 through the uses of virtual printer devices, thus enabling the PDSC 192 to be compatible with all applications 192 and formats. Importantly, because the PDSC 192 is integrated directly with portable document technology, the sender 16 of a document need not make assumptions about the capabilities of the intended recipient 22 of the document.

The PDSC 192 allows two primary modes of usage: print or "drag and drop". By print, a sender 16 can simply select the print option from any application 190 and trigger the sequence of events to generate a portable document, and then address and send that document. From the user's perspective, they simply select the print command and are then prompted for the destination of the document using standard addressing interfaces and address books. A Microsoft Mail™ user, for example, would be prompted with the standard Microsoft Mail™ addressing dialog to direct where a document may be sent. After selecting the destination of the document, the PDSC 192 automatically connects to a BFD server 12 and securely uploads the documents 166 and the intended list of recipients 22, as well as any other attributes selected to customize the send. "Drag and Drop" usage allows users 16 to avoid launching applications and printing to send documents; the document may simply be dropped on a PDSC 192 send icon, which is accessible from the sender's desktop 164.

Additional functionality and customization is one click away. During the addressing process, users 16 are free to customize the options of their send by invoking advanced options. By default, each send will reuse the existing parameters for sending documents. Users 16 can also use the advanced options user interface 193 to customize their delivery options, including, for example, security options and receipt requirements. For example, if the user 16 desires to customize the security options, including private and or public key encryption, the user simply checks a "Public Encrypt" or "Private Encrypt" option. Similarly, the user can select the "Notify on Receipt" option, thus informing the BFD server 12 to confirm delivery when the document is actually received.

BFD Server Configuration Options and User Interface. The BFD Server 12 can be configured and customized directly from a sender desktop 164. The access to the BFD server 12 from the desktop is achieved using an HTML forms user interface. This user interface exists to give server administrators access and control over the advanced options of the BFD server 12. For example, a server administrator might update the database of the 100,000 recipients who are intended to receive a specific document, and then directly invoke the send of the document to those recipients. The server administrator might generate a report regarding the send transactions which occurred during the previous week.

To access the BFD server 12 from the desktop 166, a user 16 must have a special account created on the BFD server 12, which is created ahead of time by the BFD server 12. Additionally, accessing the BFD server 12 over this account requires several layers of authentication and security, thus preventing unsolicited access.

The Server Configuration User Interface 198 allows the user 16 to access and control the server settings, which may include transaction management, account management, reporting facilities, direct upload and download of documents to distribute, direct manipulation of recipient lists, and direct access to send options.

Portable Document Receive Client. The recipient 22 of a document can utilize the portable document receive client (PDRC) 194 to access and manipulate documents which were sent to the recipient 22 by the portable document send Client 192 or by the BFD server 12 directly via the BFD server administrator. In the event that the recipient 22 of a document does not already have a PDRC 194, the software may be downloaded and installed directly from the Internet. The architecture of the portable document delivery system 160 simplifies this process, and employs dedicated software and scripts, in addition to advents in new browser architectures to enable first-time recipients 22 to be one click away from accessing the necessary software to receive documents.

The most basic case of the portable document receive client 194 can simply function as browser extension, such as a Netscape NAVIGATORT™ plug-in or a Microsoft ActiveX™ control. For other users, the PDRC 194 will behave as

a stand alone application which works as a helper application.

A third application exists for portable document delivery system 160 customers who prefer direct access to the portable documents from the recipients desktop 170. In this configuration, a dedicated portable document receive client 194 can be downloaded directly from the Internet. This component will continually monitor the activity of the portable document delivery system 160, and will automatically extract any incoming portable documents from BFD servers 12, and open them for immediate document communication on the computer desktop 170 of the recipient 22.

Recipients 22 of portable documents from the portable document delivery system 160, depending on the send configuration options, will be allowed to view, search, print, archive, or export information from their documents. Documents distributed using Envoy™ or Acrobat™ in conjunction with the portable document delivery system 160 will preserve complete visual fidelity and may be produced on high resolution output devices with the highest level of quality.

Figure 20 illustrates how a document can be sent by the fax gateway 56 of a BFD server 12 to a printer 178. Figure 21 illustrates how a document can be sent by the department gateway 202 of a dedicated corporate BFD server 200 through a LAN 204 to a department printer 178.

Private, Trackable URLs for Directed Document Delivery. This embodiment of the invention provides a unique means of delivering documents electronically. Importantly, this embodiment of the invention enables a number of value added services, in addition to basic document delivery, including but not limited to tracking and security.

The invention provides a document delivery architecture which dynamically generates a private Uniform Resource Locator (URL) to distribute information. Each private URL ("PURL") uniquely identifies the intended recipient of a document, the document or set of documents to be delivered, and (optionally) other parameters specific to the delivery process. The intended recipient of a document uses the PURL to retrieve a document (or documents). The server, upon retrieval of the document, customizes the behavior of the retrieval based upon attributes included in the PURL, as well as log information associated with the retrieval in a data base. This architecture and usage of PURLs enables secure document delivery and tracking of document receipt.

The World Wide Web ("Web") enables consumers to retrieve content from Web servers using Web browsers. In short, consumers pull content from the Web. E-mail enables producers of content to send that content to consumers. In other words, producers push content with e-mail. E-mail Internet servers, as well as the SMTP protocol (simplified mail transport protocol) which governs the behavior of Internet servers, are limited capabilities they provide to users of the Internet. For example, SMTP e-mail servers do not know anything about binary file types, tracking, or security.

The Web and the associated HTTP protocol, by contrast, provides a flexible protocol that enables the efficient, secure transmission of binary information. HTTP, however, is a pull, consumer driven protocol, and hence a producer or sender of information cannot rely on HTTP exclusively to direct the delivery of information.

By combining HTTP for the delivery, as well as using SMTP/e-mail for notification, it is possible to build a solution that allows the producer to be the driver, or to push, but that does not suffer from the limitations and legacy issues associated with SMTP/email.

PURLs are temporary, dynamically generated uniform resource locators which uniquely identify the intended recipient of a document and the document itself, as well attributes associated with the delivery of a document. PURLs avoid attaching information to e-mail messages to send documents, but rather attach a general reference to a document to be sent, and then enable the recipient to access a document via the reference.

When the recipient accesses the document by using the reference, a server can intercept the request to access the document and provide value added services, such as tracking and security. For example, a user can include a key in the PURL that serves to unlock a document on a server, perhaps decrypting an encrypted document. Or, a user can include a unique identification number in the PURL that identifies the recipient. In this case, the server can notice that a specific individual has accessed a specific document, can note that in a data base, and can make that information available to the sender. This embodiment of the invention can therefore provide document tracking.

Figure 22 is a block diagram which depicts a document delivery system that includes private, trackable URLs for directed document delivery according to the invention. A document 310 is forwarded from a sender 300 to a server 315. The server temporarily stores the document. The server dynamically generates a URL for each intended recipient of the document. In addition to encoding user information and document information with the URL, the server also encodes delivery parameters, or transaction identifiers in the URL. Each generated personal URL (PURL) is then forwarded to each intended recipient 320. The recipient is notified 325 that a given document has been sent to him. This typically has the form of an e-mail message which includes a private URL. The recipient, using the PURL 330 and the Web, accesses the document.

When the recipient accesses the document via the PURL, the recipient presents the PURL to the server. The server then has the opportunity determine the next set of actions. For example, the server could notice that the PURL specifies that a password must be presented before the electronic document referenced by the PURL can be accessed. The server may also identify the specific recipient accessing the document by the PURL, and log the fact that the specific recipient has attempted access the specific document, again all identified by the PURL. The server may also log the fact that the entire document was delivered successfully.

Accordingly, a data base maintained on the server has a full log describing the following, for example:

- Who accessed the document;
- When they accessed the document; and
- Whether they successfully accessed the document.

This information which the server has logged can then be reported back to the sender of a document. Hence, using a combination of e-mail for notification, the Web for delivery, and private URLs to identify recipients and documents, a delivery server can be constructed to track documents and report the delivery state of a document back to the sender. The actual implementation of such system may be in accordance with the system herein described in connection with Figures 3-21, or it may take other forms as appropriate.

In other embodiments of the invention, the server can log other types of information. Thus, the server can log the IP address associated with a given recipient who is retrieving a document. The server can also log the IP address of any subsequent accesses to a given document with the same PURL. Thus, the server could prevent multiple IPs from accessing the same document using the same key. Alternatively, the server could provide a list to the sender containing IP addresses which accessed a specific document intended for a specific recipient.

The above described architecture for delivery also facilitates security. A document can remain encrypted on the server until a recipient presents a valid key to access and decrypt a document. This key is presented as encoded in part of the PURL. Alternatively, the PURL specifies that a key must be retrieved, in which case the server requires that the recipient present a unique password to decrypt the document. In the first case, retrieval of the encrypted document is a one-step, automatic process because the key is encapsulated in the PURL.

PURL Implementation.

First, consider the potential construction of a PURL. The following diagram outlines one specific example of a PURL:

http://posta.tumbleweed.com/cgi/posta.dll? pu = 0-233-33982-FIAAAV4

The above PURL denotes the following:

Value	Meaning
http:/	Use the HTTP protocol to access.
posta.tumbleweed.com	Name of the HTTP server.
cgi/posta.dll	Name of HTTP server extension.
pu=0	Don't use a password.
233	Store item Identifier.
33982	Recipient Identifier.
FIAAAV4	Key to access the document.

With further reference to Figure 22, it should be noted that a PURL 302 is shown having various fields. These fields include a password identifier 331, a store item identifier 332, a recipient identifier 333, a document key 334, and any other optional fields that may be desired 335. These fields are discussed in greater detail below.

Password Identifier. A password identifier specifies whether a password is required to access a given document. In this case, the value "0" indicates no password is required. A value of "1" indicates a password is required.

Store Item Identifier. A store item identifier uniquely identifies which document a given recipient desires to obtain. In this case, the value "233" provides an index into a sparse table on the server, identifying a value which, *e.g.* identifies where a given document resides on the server and/or what a document is named.

Recipient Identifier. A recipient identifier uniquely identifies the intended recipient of a given document. In this case, the value "33982" provides an index into a sparse table on the server. The value at this table index contains recipient information.

Document Key. The document key validates the PURL itself. In this case, the key is a randomly generated number

associated with the given recipient and store identifiers. The key is used to validate whether the given recipient identification number is valid, whether the given store identification number is valid, and whether the given recipient with the given store identification number should be granted access to a document. In other embodiments of the invention, the key also encodes an index into a table which contains the validation information, as opposed to encoding the validation information itself.

Importantly, the server has a Web extension, enabling the HTTP processing of a document to be extended to provide customization. Thus, the recipient accessing the document goes through an HTTP server extension to communicate with an HTTP server. This extension, for example, can decide to grant access to a document, in which case it presents the user with a new PURL which facilitates transmission of the specific document.

The server can use the above attributes and values of a PURL to customize the behavior of document delivery. Specifically, the server executes the following steps to deliver the document and record the delivery transaction:

- Decode the PURL into its various parts;
- Validate each component of the PURL;
- Authenticate the PURL using the key;
- Determine which user is accessing the document by using the Recipient
- Determine which document the user is accessing by using the Store Item
- Determine whether the document, given the above, requires additional input
- Deliver the document to the recipient;
- Log all attributes of the transaction, including, *e.g* time of access, success of transmission, and IP of recipient.

Once information has been logged in a data base running on the server which records transaction information, this data can be accessed by the recipient and can even be dynamically transmitted back to the recipient. For example, a given publisher (sender) asks the server's data base for all documents which have been delivered to a specific recipient. The publisher asks the server to generate a report of the status of a given document sent to ten people. The server reports back, for example, that the document has been sent to all ten people at a specific time, but only three of the people have actually retrieved the document. Each document retrieval may include the specific time the document was accessed, the time it was accessed, and whether it was accessed completely and successfully. Hence, dynamically generated PURLs as broadcast over email enable a robust means of tracking the delivery of documents over wide area networks.

Although the electronic document delivery system and its methods of use are described herein in connection with use in the Internet, the invention may be applied to any of a wide variety of networks, including internets, intranets, LANs and WANs, or any combination thereof, as desired. As well, the invention may be applied to a wide variety of computer platforms, communication protocols, portable document formats, or any combination thereof, as desired.

The invention provides a method and system for secure document delivery over a wide area network. A document Delivery Server dynamically retrieves a public key of an intended recipient of a document, then uses the public key to encrypt either a document or the secret key of the document. The server delivers the encrypted document to an intended recipient over a wide area network such as the Internet. The intended recipient decrypts the document using the private key associated with the public key. The invention permits only an intended recipient to gain access to a specific document and therefore provides a unique level of security for document delivery.

For the purposes of the invention, the term document includes any contiguous collection of data, including a stream of data, a video, audio data, an animation, a formatted document such as HTML, PDF, or Envoy, or a data base. While the preferred embodiment of the invention is adapted for use in document transmission over the Internet, the invention is equally applicable to other wide area networks.

Furthermore, while the preferred embodiment of the invention discloses transmission of a document to a recipient computer, the invention is operable for document transmission to any intended recipient maintaining, or having the ability to dynamically generate, a private/public key and to use the private key to decrypt a document encrypted with the corresponding public key. An intended recipient, therefore, includes, for example, an Internet user of a desktop computer, printer, fax machine, personal digital assistant, or network computer device.

Similarly, while the sender of a document is preferably a desktop computer, the sender also includes any device capable of encrypting a document and communicating with the Delivery Server, such as a network computer device. In

an alternative embodiment of the invention, the document is encrypted by the delivery server. In this embodiment, the sender includes any device, such as an Internet browser device, Internet telephone device, personal digital assistant, or fax machine, that can transmit a document to the Delivery Server for encryption and transmission to the intended recipient.

Figure 23 is a diagram of a system for dynamic server document encryption, according to a first preferred embodiment of the invention. A document stored on a desktop computer, the sender 1032, is to be transmitted to another computer, the intended recipient 1034. In this first preferred embodiment, the document is stored in Portable Document Format (PDF). However, in alternative embodiments, a document may be stored in any appropriate format. Portable Document (PD) formats are required for distributed print and fax solutions. However, PD formats are not required for the invention.

The document is sent from sender to recipient via the Delivery Server 1036. In this first preferred embodiment of the invention, the Delivery Server is directed by the sender to communicate with a certificate authority database server 1038 to retrieve the intended recipient's public key (certificate). The Delivery Server dynamically queries the certificate authority and retrieves the public key. The public key is transmitted to the Delivery Server and from there to the sender. In alternative embodiments of the invention, the Delivery Server retrieves the intended recipient's public key from the intended recipient's desktop computer, an Internet server, or from an intranet server connected to the intended recipient's desktop computer.

In the first preferred embodiment of the invention, the sender encrypts the document using a secret key and uses the public key to encrypt the secret key. The document and encrypted secret key are then transmitted to the intended recipient. The secret key is decrypted with the intended recipient's private key and is then used to decrypt the document.

In an alternative, equally preferred embodiment, the sender uses the public key to encrypt the document. The encrypted document is then transmitted to the intended recipient and decrypted using the private key associated with the public key.

Figure 24 is a flow chart of the set of operations for dynamic server document encryption, according to a first preferred embodiment of the invention. In the example, the sender encrypts the document 1040 using a secret key. Such secret key includes any appropriate encryption scheme known in the prior art. The sender then contacts a Delivery Server 1045 to query 1050 the public key associated with the intended recipient. The Delivery Server retrieves this certificate in real time 1055, for example from the data base of a certificate authority, and transmits the certificate back to the sender 1060.

In the event that the certificate authority returns no certificate, the Delivery Server dynamically generates a new certificate for the recipient. To do so, the Delivery Server forwards a dynamically generated URL in an e-mail message to the recipient. Recipient access of the URL dynamically retrieves a Java Applet or Plug-in, which is automatically downloaded to the recipient's system. This applet or Plug-in then runs on the Recipient system and constructs a private/public key pair. Generating a private/public key pair on a local machine is not specific to this invention and is documented in a number of sources.

The applet or plug-in next forwards the public key to the Delivery Server. The server, using properties of the generated URL, identifies the e-mail address of the recipient. Thus, the generated public key has the property of having authenticated the e-mail address of the recipient, as the URL to invoke the key generation has only been forwarded to a specific e-mail address. The server combines the e-mail address and public key into a certificate and returned to the Send Client or used by the server to encrypt the document or secret key. The Delivery Server, using LDAP or a similar protocol, may communicate the certificate to the certificate authority. Alternatively, the Delivery Server simply may maintain a local database or dynamically generated certificates for future use.

Upon receiving the public key from the Delivery Server, the sender encrypts the secret key 65 with the public key. In an alternative, equally preferred embodiment of the invention, the sender does not encrypt the document until the public key has been received. Because the document is not encrypted if the public key is not authenticated, this embodiment minimizes processing time when a public key cannot be retrieved.

The sender then forwards 1070 the encrypted document, the address of the intended recipient (for example an email address), delivery instructions, and the encrypted secret key to the Delivery Server over a secure channel. Thus, the document does not leave the Sender until the document has been encrypted with the secret key and the secret key has been encrypted with the intended recipient's public key. The Delivery Server then delivers 1075 the encrypted document and secret key to the intended recipient. The intended recipient, using the private key associated with the public key, decrypts the secret key 1080 and uses the secret key to decrypt the document. Such scheme prevents unauthorized access to the document, since the document can only be accessed by the owner of the public key.

Figure 25 is a flow chart of the set of operations for dynamic server document encryption, according to an alternative embodiment of the invention. The sender notifies the Delivery Server 1090 that the sender intends to send a document to a given recipient. The Delivery Server queries 1095 the certificate authority to obtain the intended recipient's public key, which is returned 1100 to the Delivery Server.

In this embodiment, the Sender does not encrypt the document but forwards the document 1105 to the Delivery Server over a secure channel. The Delivery Server then encrypts 1110 the document using a secret key. The Delivery Server uses the retrieved public key of the intended recipient to encrypt 1115 the secret key, and then forwards the encrypted document and secret key to the intended recipient 1120. The intended recipient uses the private key to decrypt the secret, and then uses the secret key to decrypt the document 1125.

Alternatively, the Delivery Server may use the public key to encrypt the document. The encrypted document is then transmitted to the recipient.

In the preferred implementation of the invention, the sender is connected to the intended recipient via a Delivery Server, all running over a wide area network, such as the Internet. The sender is preferably a computer using software referred to herein as the Send Client.

The Delivery Server is responsible for determining the public key of a given recipient and forwarding that key to the Send Client. The Delivery Server is also responsible for delivering the encrypted document and secret key to the intended recipient.

The Send Client initiates the delivery transaction by first identifying the document to be delivered, any delivery parameters, and the set of intended recipients to receive the document. Delivery parameters include such options as the scheduled delivery time, security options, urgency of the delivery, presentation parameters for the delivery, and receipt notification.

The Send Client then initiates a dialog with the Delivery Server and encrypts the document with a secret key. The dialog and encryption steps may be performed simultaneously or sequentially, depending upon the sender's hardware and software configuration. In the dialog, the Send Client forwards to the Delivery Server the intended recipient(s) of a given document. The Send Client requests that the Delivery Server contact the Send Client once the public key has been acquired.

The Send Client expresses the identity of the intended recipient(s) of a given document in different ways. In the preferred embodiment of the invention, the Send Client uses the electronic mail (email) address of the intended recipient as the identifier of the intended recipient. However, the Send Client can also identify the intended recipient with an alternative identifier, such as a driver's license number, a social security number, an abstract identifier, a symbol name, or a fax number.

The Delivery Server uses several techniques to obtain the certificate for the intended recipient. In the preferred embodiment of the invention, the Delivery Server contacts a certificate authority data base server, presents information identifying the intended recipient, and asks for the intended recipient's public key. The invention may therefore be used to obtain information from certificate authorities that maintain public key data bases that can be accessed dynamically over a programmatic interface (queried) in real time.

The invention is implemented using any appropriate means for a Delivery Server to query a public key of an intended recipient in real time without user intervention. Thus, the specific protocol and means of accessing the public key data base are not significant for the invention. The public key data base is preferably accessed using the Internet Lightweight Directory Access Protocol (LDAP) standard developed by the University of Michigan in conjunction with the Internet Engineering Task Force. LDAP servers provide directory and other services. Using LDAP protocol, a given server may be queried, and information maintained on that server may be retrieved over an electronic network. LDAP servers can be queried directly using standard Internet protocols. Alternative embodiments of the invention use, for example, SQL Queries with different connectivity protocols including RPC (remote procedure call).

The certificate authority data base server and the Delivery Server may be either the same or separate servers. Maintaining both the certificate authority data base and the Delivery Server on the same server is advantageous for a dedicated application of document delivery which does not require access to a general data base of certificates. For example, a corporation may maintain a database of employees' public keys on the same server used for Internet communications. The same server is therefore used as the certificate authority data base and as the Delivery Server for interoffice communication within the company.

For embodiments in which the certificate authority data base server and the Delivery Server are separate, the Delivery Server may maintain a cache or local copy of recently queried certificates. Use of such cache saves time in future queries for the same recipient and certificate.

The invention supports document delivery to one or more recipients. For multiple recipients, the process discussed above is applied in batch mode. An ordered list of intended recipients is forwarded to the Delivery Server, and the Delivery Server returns a corresponding ordered list of certificates.

The invention may also be used to send multiple documents from sender to recipient(s). In such case, a single secret key is used to encrypt each document. Once the Delivery Server has returned a certificate containing each recipient's public key, the single secret key is encrypted with the retrieved public key(s) of the intended recipient(s). For each recipient, the Send Client forwards an encrypted secret key and the encrypted document(s) to the Delivery Server, along with the intended recipient address and delivery parameters.

The Delivery Server then forwards to each recipient the combined encrypted secret key and document(s). The

recipient device uses software known as the Receive Client. The Receive Client is currently implemented as a Java Applet as well as a plug-in to standard internet browsers. Java is a programming language developed by Sun Microsystems of Mountain View, CA. However, the Receive Client may also be implemented using any other programming language that is capable of receiving and decrypting the transmitted secret key and document(s).

When implemented as a Java Applet, the Receive Client is distributed dynamically from the Delivery Server to the intended recipient's system. The Receive Client uses the private key to decrypt the secret key. This decrypted secret key is then used by the recipient to decrypt the document(s).

In the preferred embodiment of this invention, the Receive Client accesses the encrypted secret key and document from the Delivery Server using Hypertext Transmission Protocol (HTTP), the standard internet delivery protocol. However, the Receive Client may access the Delivery Server using any other appropriate protocol.

When using HTTP, the Receive Client is sent a uniform resource locator (URL) containing the address of the documents and key to be delivered. In the preferred embodiment of the invention, the document(s) and secret key are packaged into a single file or stream of data, which is delivered intact to the Receive Client using HTTP. The Receive Client is thereby given maximal flexibility to retrieve the package and decrypt it from the recipient(s) web browser. The recipient may use any web browser or other software application that is capable of receiving the data transmitted over the wide area network.

Although the invention is described herein with reference to the preferred embodiment, one skilled in the art will readily appreciate that other applications may be substituted for those set forth herein without departing from the scope of the present invention.

The source code for the Send Client, the Receive Client, and for the Delivery Server software can be readily configured by one skilled in the art using well-known programming techniques and hardware components. Additionally, Send Client and Delivery Server functions may also be accomplished by other means, including integrated circuits and programmable memory devices such as an EEPROM.

The implementation of the dynamic server document encryption discussed above with regard to the preferred embodiment of the invention is only one possible implementation. Alternate embodiments may use other implementations consistent with the teachings of the invention.

The Receive Client may be configured to direct a document to another device. For example, a decrypted document may be sent to a printer or a fax machine.

The invention may use any appropriate encryption scheme for the secret key, public key, and private key, including the RSA and Verisign schemes.

Although the present invention has been described in detail with reference to a particular preferred embodiment, persons possessing ordinary skill in the art to which this invention pertains will appreciate that various modifications and enhancements may be made without departing from the spirit and scope of the claims that follow.

Claims

1. An apparatus for delivering an electronic document between a sending computer (16) and a receiving computer (22), comprising:

a server (12) interposed between said sending computer (16) and said receiving computer (22), wherein when said electronic document is forwarded to said server (12) from said sending computer (16) and said server (12) dynamically generates a private Uniform Resource Locator ("PURL") to distribute said electronic document.

2. The apparatus of Claim 1, wherein said PURL uniquely identifies an intended recipient (22) of said electronic document and, optionally, other parameters specific to said electronic document delivery.

3. The apparatus of Claim 2, wherein said intended recipient (22) of said electronic document uses said PURL to retrieve said electronic document.

4. The apparatus of Claim 3, wherein said server (12), upon retrieval of said electronic document, customizes the behavior of said retrieval based upon attributes included in said PURL and, optionally, log information associated with said retrieval in a data base, to enable secure document delivery and tracking of document receipt.

5. The apparatus of one of the preceding claims, wherein said server uses HTTP for delivery of said electronic document and SMTP/e-mail for notification of arrival at said server of said electronic document.

6. The apparatus of one of the preceding claims, said PURL comprising:

a temporary, dynamically generated uniform resource locator which uniquely identifies an intended recipient (22) of said electronic document and, optionally, said electronic document itself and attributes associated with delivery of said electronic document.

7. The apparatus of one of the preceding claims, wherein said PURL attaches a general reference to an electronic document to be sent, and enables a recipient (22) to access said electronic document via said reference.

8. The apparatus of Claim 7, wherein said server (12) intercepts the reguest to access said electronic document and provides a value added service in connection with said access, when said recipient (22) accesses said document by using said reference.

9. The apparatus of one of the preceding claims, said PURL further comprising:

a key that unlocks a document on said server (12).

10. The apparatus of one of the preceding claims, said PURL further comprising:

a unique identification number that identifies a recipient of said electronic document.

11. The apparatus of Claim 10, wherein said server (12) notices that a specific individual has accessed a specific document and notes that in a data base to provide document tracking.

12. A document delivery system for delivering an electronic document (310) between a sender (300) and at least one recipient (320), comprising:

a document server (315) that temporarily stores said electronic document (310), wherein said server (315) dynamically generates a private, trackable URL ("PURL") (330) for each intended recipient (320) of said document (310) that is forwarded to each intended recipient (320).

13. The system of Claim 12, wherein said server (315) encodes delivery parameters, or transaction identifiers in said PURL (330).

14. The system of Claim 12 or 13, wherein said PURL (330) comprises:

an e-mail message.

15. The system of one of claims 12 to 14, wherein said recipient (320) accesses said electronic document (310) via said PURL by presenting said PURL (330) to said server (315).

16. The system of Claim 15, wherein said server (315) requires that said PURL (330) specifies that a password must be presented before the electronic document (310) referenced by said PURL (330) can be accessed.

17. The system of Claim 15 or 16, wherein said server (315) identifies a specific recipient (320) accessing said electronic document (310) with said PURL (330).

18. The system of Claim 17, wherein said server (315) logs the fact that a specific recipient (320) has attempted access a specific document (310).

19. The system of Claim 17 or 18, wherein said server (315) logs the fact that an entire electronic document was delivered successfully.

20. The system of one of claims 12 to 19, further comprising:

a data base maintained on, or associated with, said server (315) that has a full log describing any of who accessed said electronic document (310), when they accessed the document (310), and whether they successfully accessed said electronic document (310).

21. The system of Claim 20, wherein information which said server (315) has logged is reported back to the sender

(300) of an electronic document (310).

22. The system of one of claims 12 to 21, wherein said server (315) can log any of the IP address associated with a given recipient (320) who is retrieving a document, the IP address of any subsequent accesses to a given document (310) with said same PURL (330), or a list containing IP addresses which accessed a specific document (310) intended for a specific recipient (320).

23. The system of one of Claims 12 to 22, wherein said electronic document (310) remains encrypted on said server (315) until a recipient (320) presents a valid key to access and decrypt said electronic document, wherein said key is presented as encoded in part of said PURL.

24. The system of one of Claims 12 to 23, wherein said PURL specifies that a key must be retrieved, and wherein said server (315) requires that a recipient (320) present a unique password to decrypt said electronic document (310).

25. The system of one of Claims 12 to 24 , said PURL (330) further comprising:

a password identifier (331) that specifies whether a password is required to access a given document (310).

26. The system of any of Claims 12 to 25, said PURL further comprising:

a store item identifier (332) that uniquely identifies which document (310) a given recipient (320) desires to obtain.

27. The system of any of Claims 12 to 26, said PURL further comprising:

a recipient identifier (333) that uniquely identifies an intended recipient (320) of a given document (310).

28. The system of any of Claims 12 to 27, said PURL further comprising:

a document key (334) that validates said PURL (330) itself.

29. The system of Claim 28, said document key (334) further comprising:

a key that is a randomly generated number associated with a given recipient (320) and a document a given recipient (320) desires to obtain, wherein said key is used to validate whether a given recipient (320) identification number is valid, whether a given store identification number is valid, and whether a given recipient (320) with a given store identification number should be granted access to a document (310).

30. A method for delivering an electronic document between a sender (300) and at least one recipient (320), comprising the steps of:

using a document server (315) to temporarily store said electronic document (310), wherein said server (315) dynamically generates a private, trackable URL ("PURL") (330) for each intended recipient (320) of said document that is forwarded to each intended recipient;

decoding said PURL into its component parts (302); and

validating each component part (302) of said PURL.

31. The method of Claim 30, further comprising the step of:

authenticating PURL (330) using a key.

32. The method of Claim 31, further comprising the step of:

determining which user is accessing a document (310) by using a recipient identifier (333) that uniquely identifies an intended recipient of a given document (310).

33. The method of Claim 32, further comprising the step of:

determining which document (310) a user is accessing by using a store item identifier (332) that uniquely identifies which document (310) a given recipient (320) desires to obtain.

34. The method of Claim 33, further comprising the step of:

determining whether said document (310) requires additional input before it can be delivered.

35. The method of Claim 34, further comprising the step of:

delivering said document (310) to said recipient (320).

36. The method of any of Claims 31 to 35, further comprising the step of:

logging all attributes of a delivery transaction, including any of time of access, success of transmission, and IP of recipient.

37. A method for secure document delivery from a sender (1032) over a wide area network, comprising the steps of:

a sender (1032) encrypting a document (1040) using a secret key (65);

the sender contacting a Delivery server (1045) to query (1050) a public key associated with an intended recipient;

the Delivery server dynamically retrieving the public key in real time (1055);

the Delivery server transmitting the public key back to the sender(1060);

the sender encrypting the secret key (65) with the public key; and

the sender (1070) transmitting the encrypted document and the encrypted secret key to the Delivery server for transmission (1075) to the recipient.

38. The method of Claim 37, further comprising the step of the recipient (1034) decrypting the secret key (1080) using a private key.

39. The method of Claim 38, further comprising the step of the recipient decrypting the document using the secret key.

40. The method of any of Claims 37 to 39, wherein the sender encrypts the document prior to receiving the public key from the Delivery server.

41. The method of any of Claims 37 to 39, wherein the sender encrypts the document subsequent to receiving the public key from the Delivery server.

42. The method of any of Claims 37 to 41, wherein the document is one of a contiguous collection of data, a stream of data, a video, audio data, an animation, a formatted document, or a data base.

43. The method of any of Claims 37 to 42, further comprising the step of the sender forwarding the address of the intended recipient and document delivery instructions to the Delivery server.

44. The method of any of Claims 37 to 43, wherein the wide area network is the Internet.

45. The method of any of Claims 37 to 44, wherein the recipient is one of a desktop computer, a printer, a fax machine, a personal digital assistant, or a network computer device.

46. The method of any of Claims 37 to 45, wherein the sender is one of a desktop computer, an Internet browser device, an Internet telephone device, or a network computer device.

47. The method of any of Claims 37 to 46, wherein the database server dynamically retrieves the public key from one of a certificate authority, an Internet server, personal digital assistant, the intended recipient's desktop computer, or from an intranet server connected to the intended recipient's desktop computer.

48. A method for secure document delivery from a sender (1032) over a wide area network, comprising the steps of:

a sender contacting a Delivery server (1036) to query a public key associated with an intended recipient (1034) of a document;

the Delivery server (1036) dynamically retrieving the public key in real time;

the Delivery server (1036) transmitting the public key back to the sender (1032);

the sender encrypting the document with the public key; and

the sender (1032) transmitting the encrypted document to the Delivery server (1036) for transmission to the recipient (1034).

49. The method of Claim 48, further comprising the step of the recipient (1034) decrypting the document using a private key.

50. The method of Claim 48 or 49, wherein the recipient is one of a desktop computer, a printer (178), a fax machine (172), a personal digital assistant, or a network computer device.

51. The method of any of Claims 48 to 50, wherein the sender is one of a desktop computer, an Internet browser device, an Internet telephone device, or a network computer device.

53. The method of any of Claims 48 to 51, wherein the database server dynamically retrieves the public key from one of a certificate authority, an Internet server, personal digital assistant, the intended recipient's desktop computer, or from an intranet server connected to the intended recipient's desktop computer.

54. A method for secure document delivery from a sender (1032) over a wide area network, comprising the steps of:

a sender contacting a Delivery server (1090) to query (1095) a public key associated with an intended recipient;

the Delivery server dynamically retrieving the public key in real time (1100);

the sender transmitting the document to the Delivery server (1105) ;

the Delivery server encrypting (1110) the document with a secret key and encrypting (1115) the secret key with the public key; and

the Delivery server transmitting the encrypted secret key and the encrypted document to the intended recipient (1120).

55. The method of Claim 54, further comprising the step of the recipient decrypting the secret key using a private key.

56. The method of Claim 55, further comprising the step of the recipient (1034) decrypting the document (1125) using the secret key.

57. The method of one of Claims 54 to 56, wherein the recipient is one of a desktop computer, a printer (178), a fax

machine (172), a personal digital assistant, or a network computer device.

58. The method of any of Claims 54 to 57, wherein the sender is one of a desktop computer, a network computer device, an Internet browser device, an Internet telephone device, or a fax machine.

59. The method of any of Claims 54 to 58, wherein the database server dynamically retrieves the public key from one of a certificate authority, an Internet server, personal digital assistant, the intended recipient's desktop computer, or from an intranet server connected to the intended recipient's desktop computer.

60. The method of any of Claims 54 to 59, further comprising the step of:

dynamically generating a public key at said Delivery server where said recipient does not have a public key at the time of said retrieval.

61. The method of Claim 60, said dynamic generating step further comprising the steps of:

forwarding a message to said recipient the reading of which retrieves a module that constructs a private/public key pair on said recipient's system.

62. The method of Claim 61, said dynamic generating step further comprising the step of:

forwarding said public key from said recipient's system to said Delivery server.

63. A system for secure document delivery from a sender (1032) over a wide area network, comprising:

a Delivery server (1036) for querying a public key associated with an intended recipient (1034) at the direction of a sender (1032), the Delivery server (1036) dynamically retrieving the public key in real time and transmitting the public key back to the sender;

the sender (1032) for encrypting a document using a secret key (65), the sender (1032) encrypting the secret key (65) with the public key and the sender (1032) transmitting the encrypted document and the encrypted secret key to the Delivery server (1036) for transmission to the intended recipient (1034).

64. The system of Claim 63, further comprising:

means for decrypting the secret key by the recipient (1034) using a private key; and

means for decrypting the encrypted document using the secret key (65).

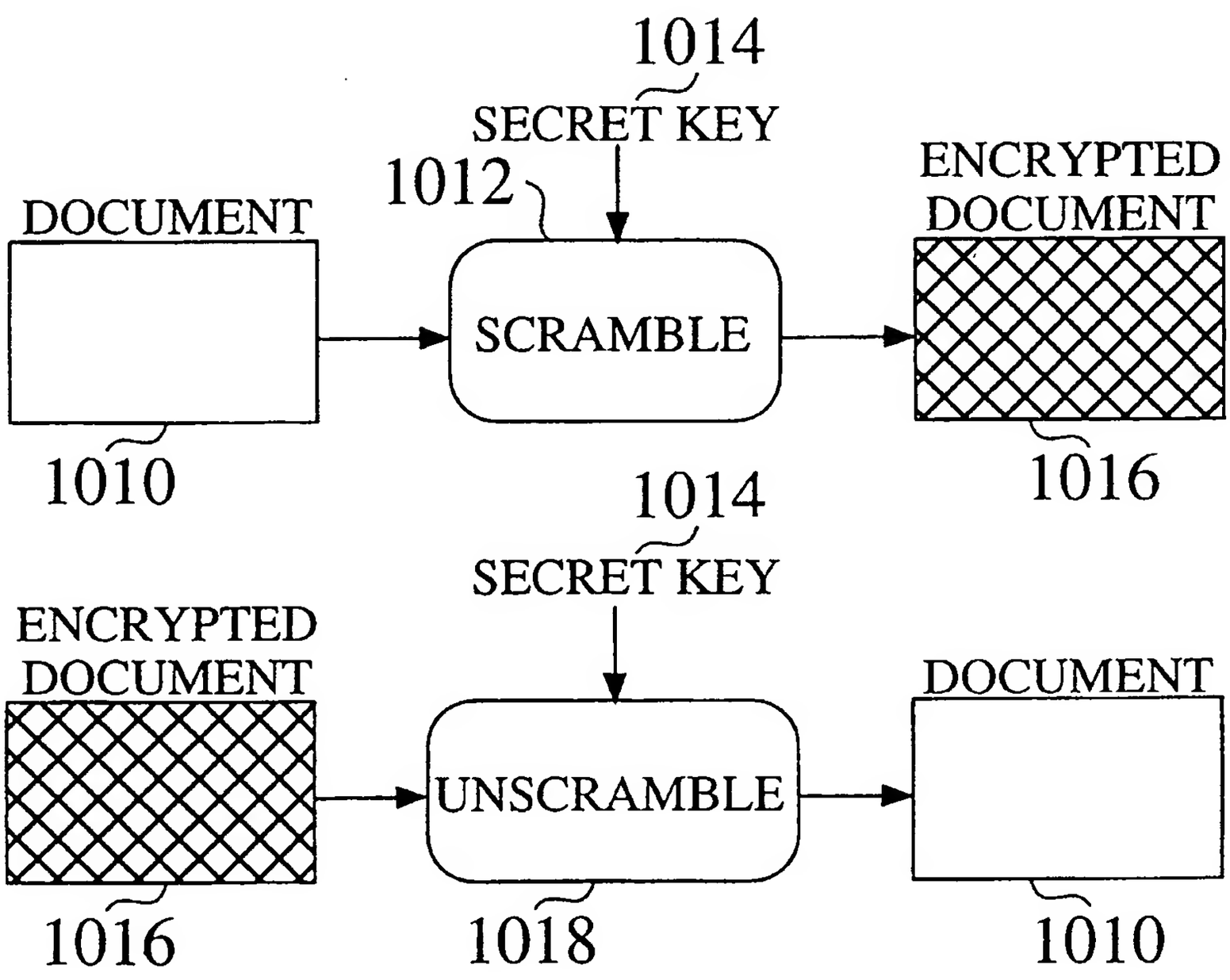

Fig. 1
(PRIOR ART)

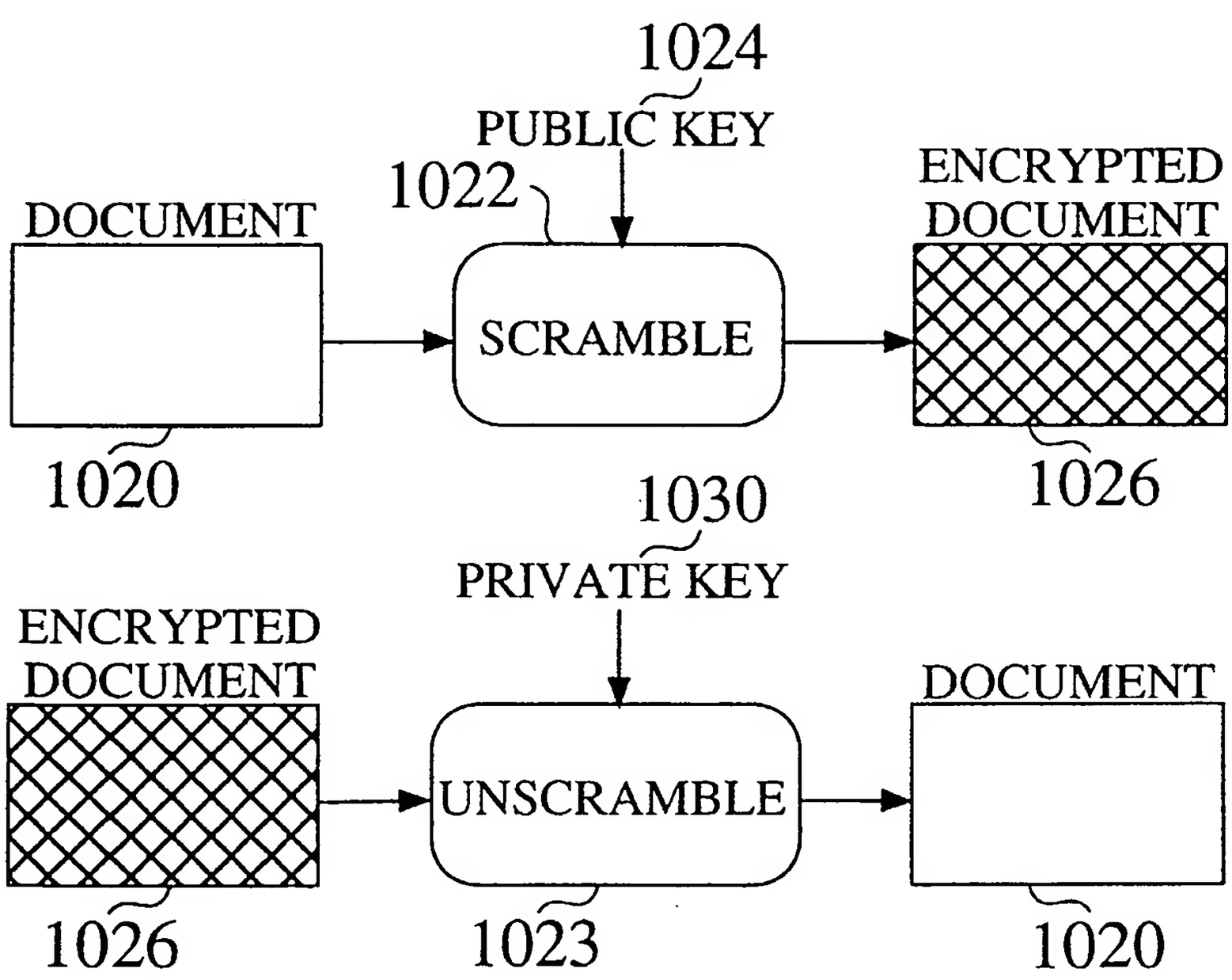
1024
PUBLIC KEY
1022
DOCUMENT
ENCRYPTED DOCUMENT
SCRAMBLE
1020
1030
1026
PRIVATE KEY
ENCRYPTED DOCUMENT
DOCUMENT
UNSCRAMBLE
1026
1023
1020

Fig. 2

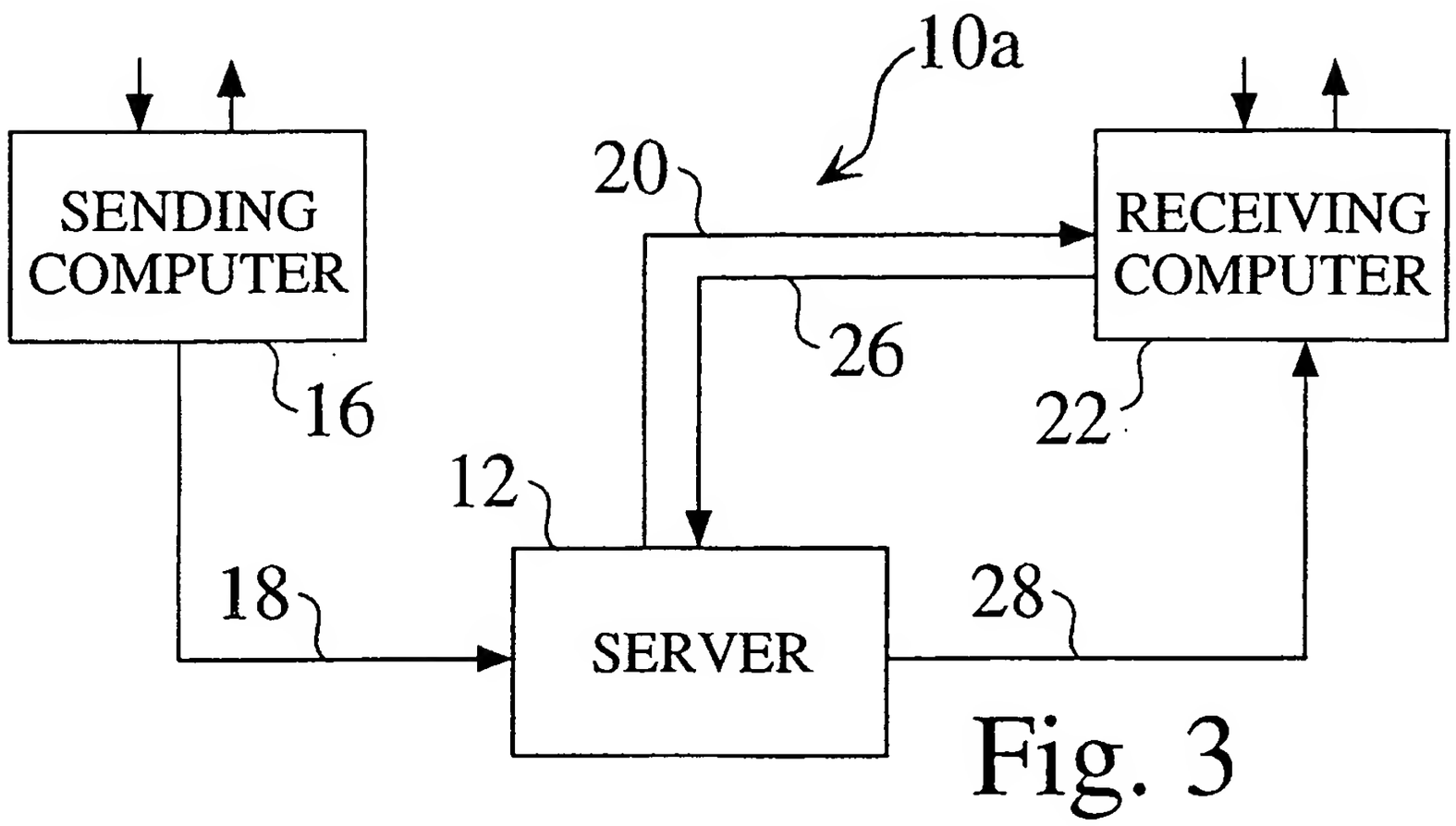

Fig. 3

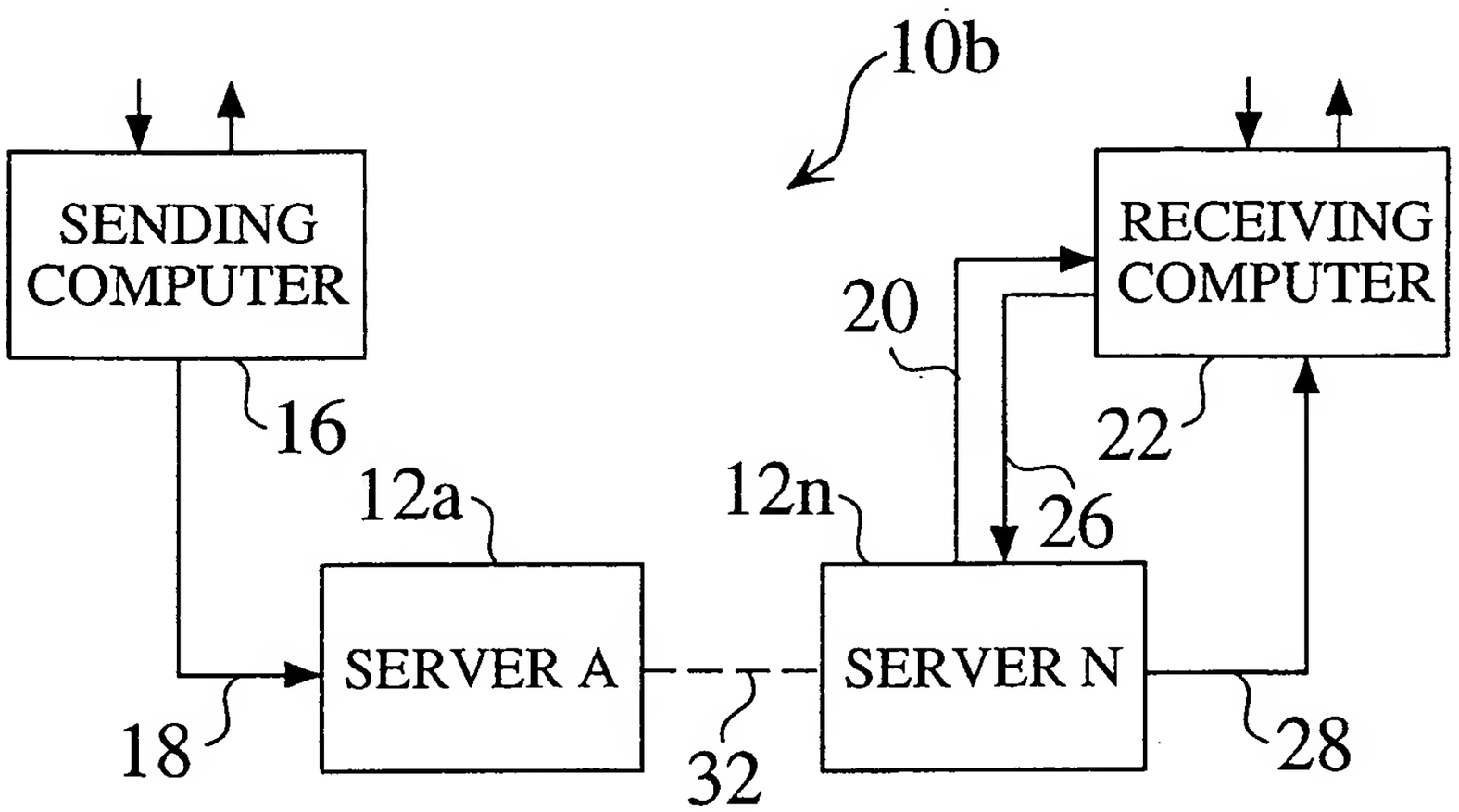

Fig. 4

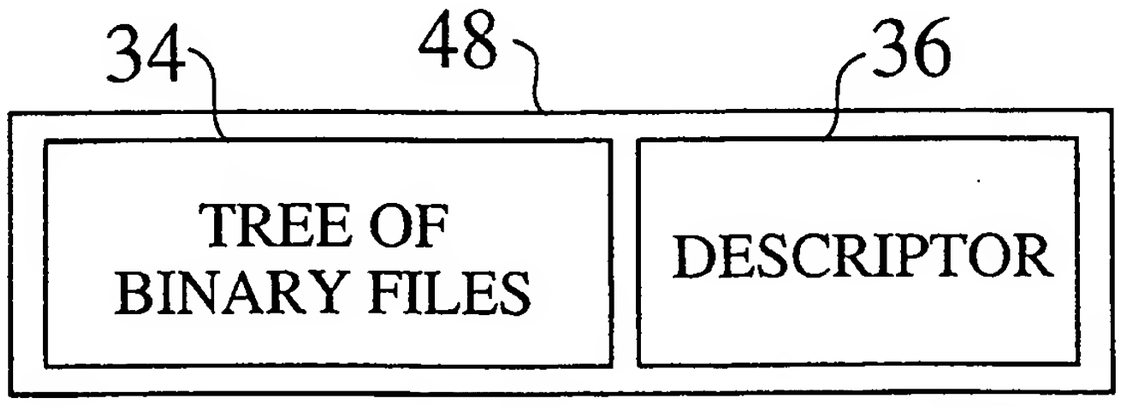

Fig. 5

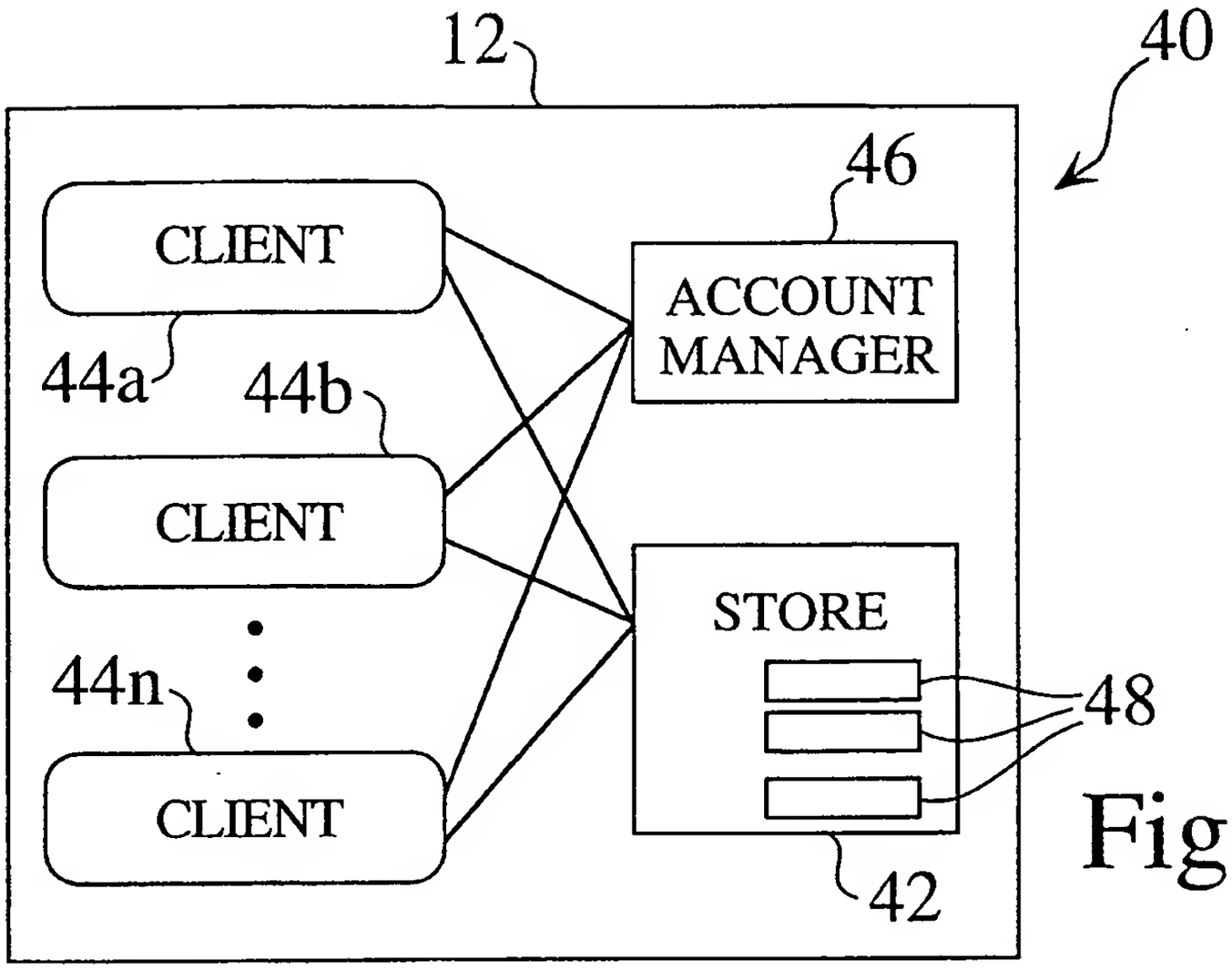
12
40
46
CLIENT
ACCOUNT MANAGER
44a
44b
CLIENT
STORE
44n
48
CLIENT
42

Fig. 6

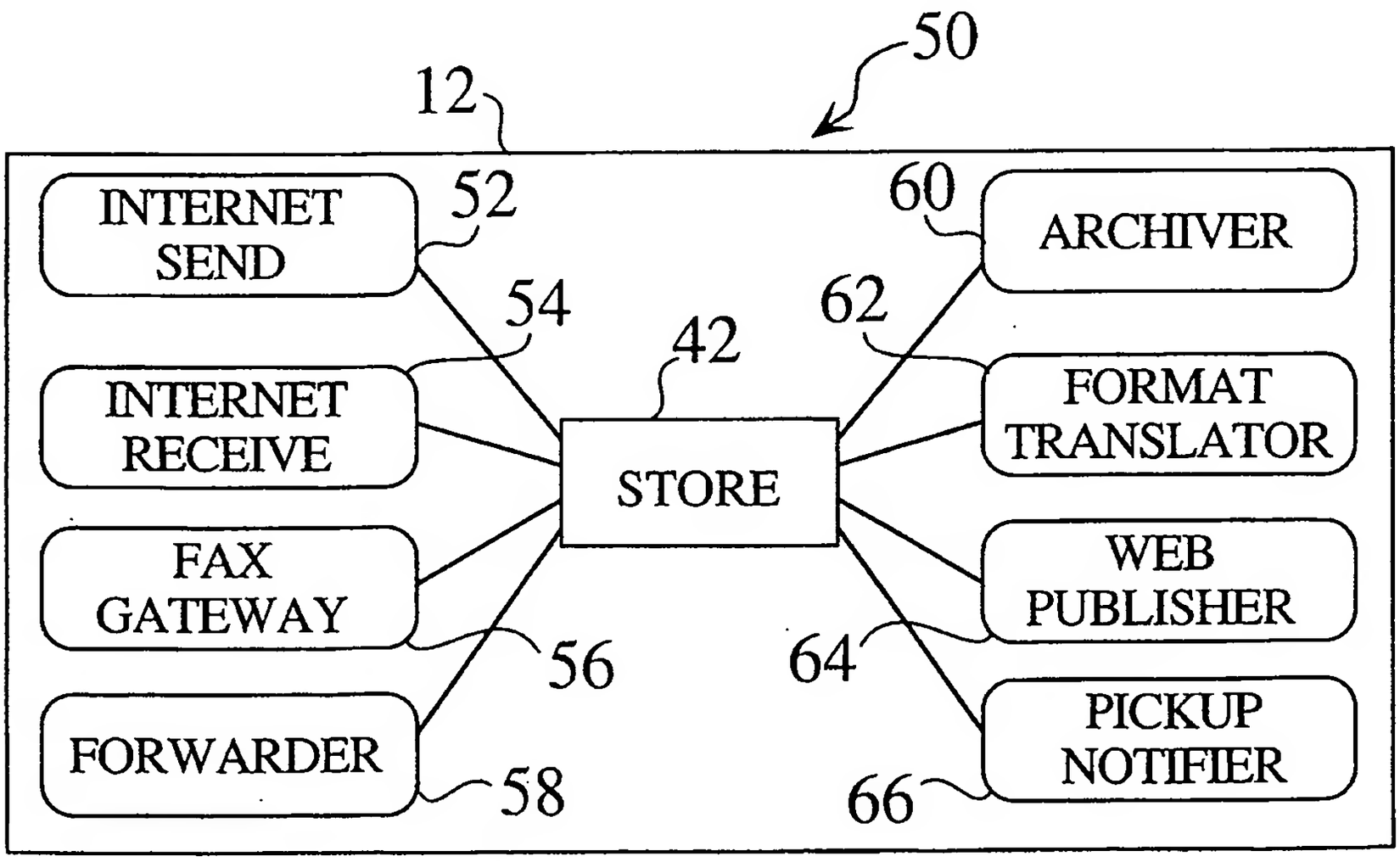
50
12
INTERNET SEND
52
60
ARCHIVER
54
62
42
INTERNET RECEIVE
STORE
FORMAT TRANSLATOR
FAX GATEWAY
WEB PUBLISHER
56
64
PICKUP NOTIFIER
FORWARDER
58
66

Fig. 7

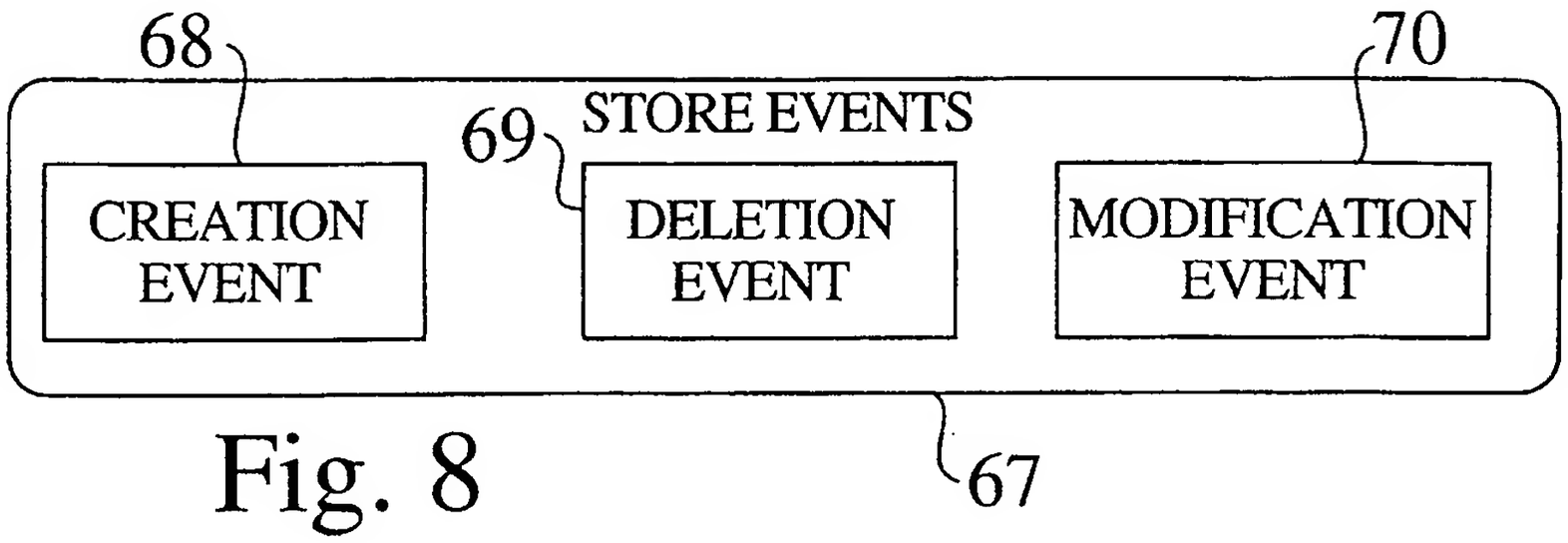
68
70
STORE EVENTS
69
CREATION EVENT
DELETION EVENT
MODIFICATION EVENT
67

Fig. 8

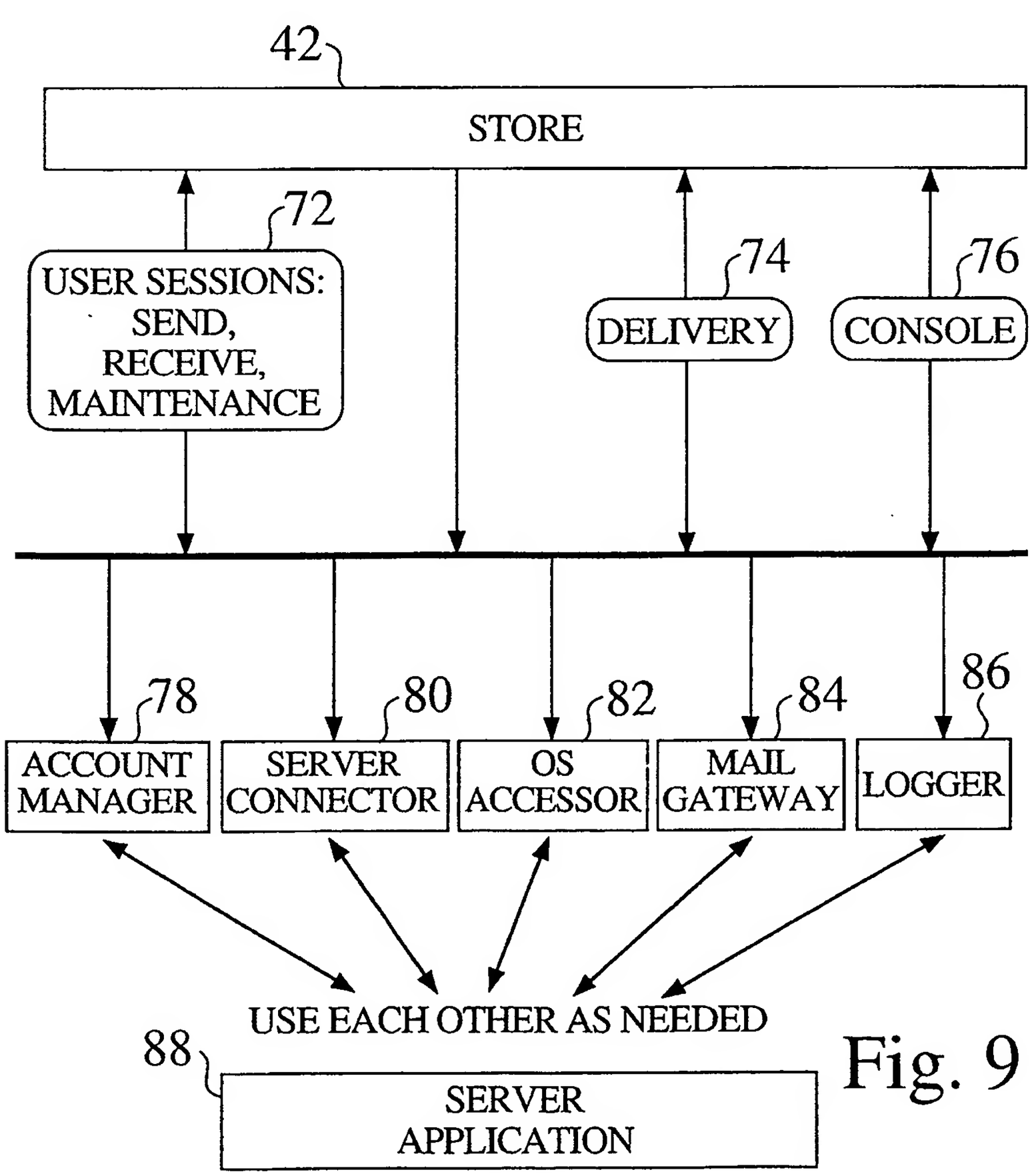
42
STORE
72
USER SESSIONS: SEND, RECEIVE, MAINTENANCE
74
DELIVERY
76
CONSOLE
78
80
82
84
86
ACCOUNT MANAGER
SERVER CONNECTOR
OS ACCESSOR
MAIL GATEWAY
LOGGER
USE EACH OTHER AS NEEDED
88
SERVER APPLICATION

Fig. 9

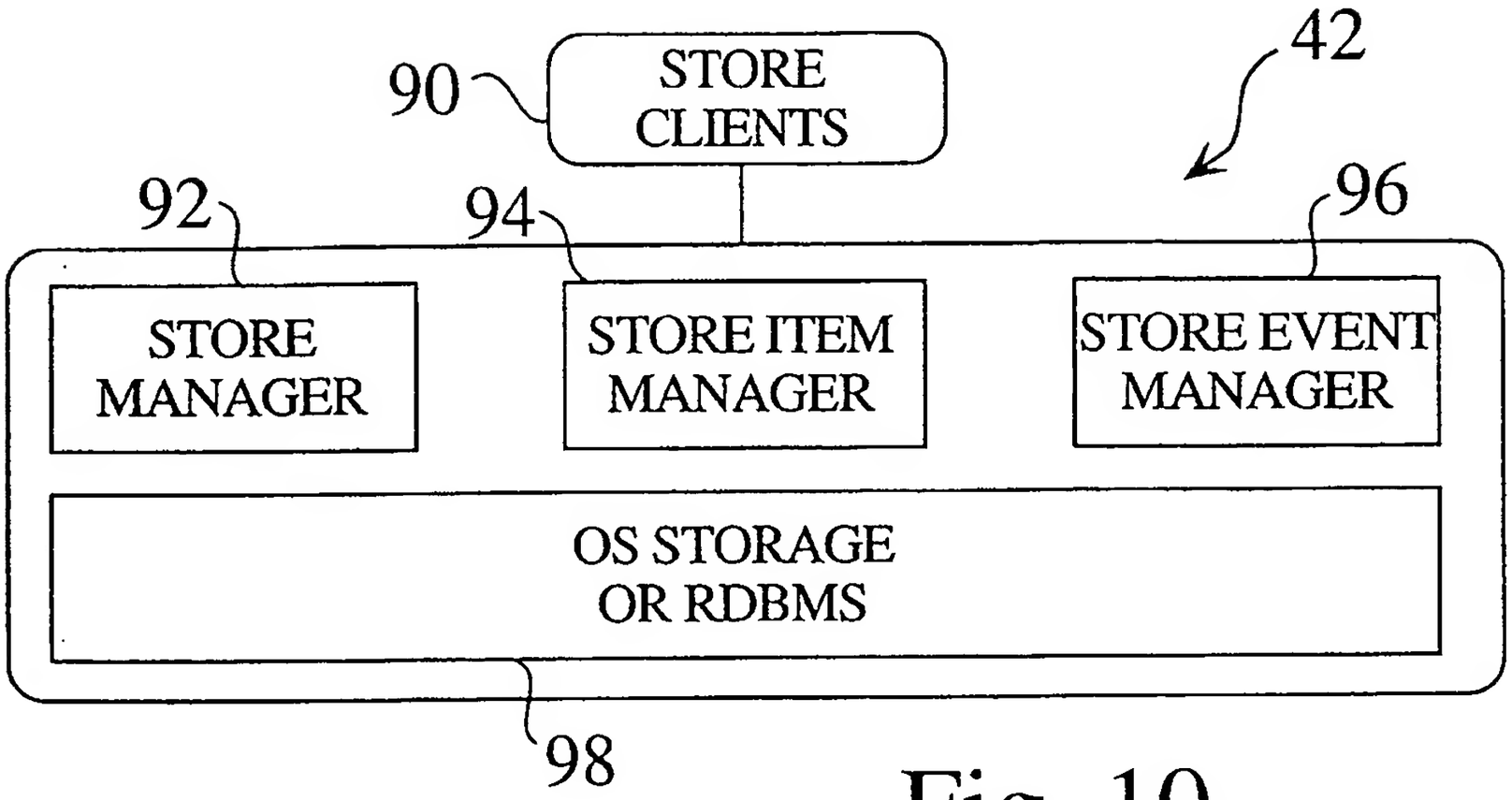
90
STORE CLIENTS
42
92
94
96
STORE MANAGER
STORE ITEM MANAGER
STORE EVENT MANAGER
OS STORAGE OR RDBMS
98

Fig. 10

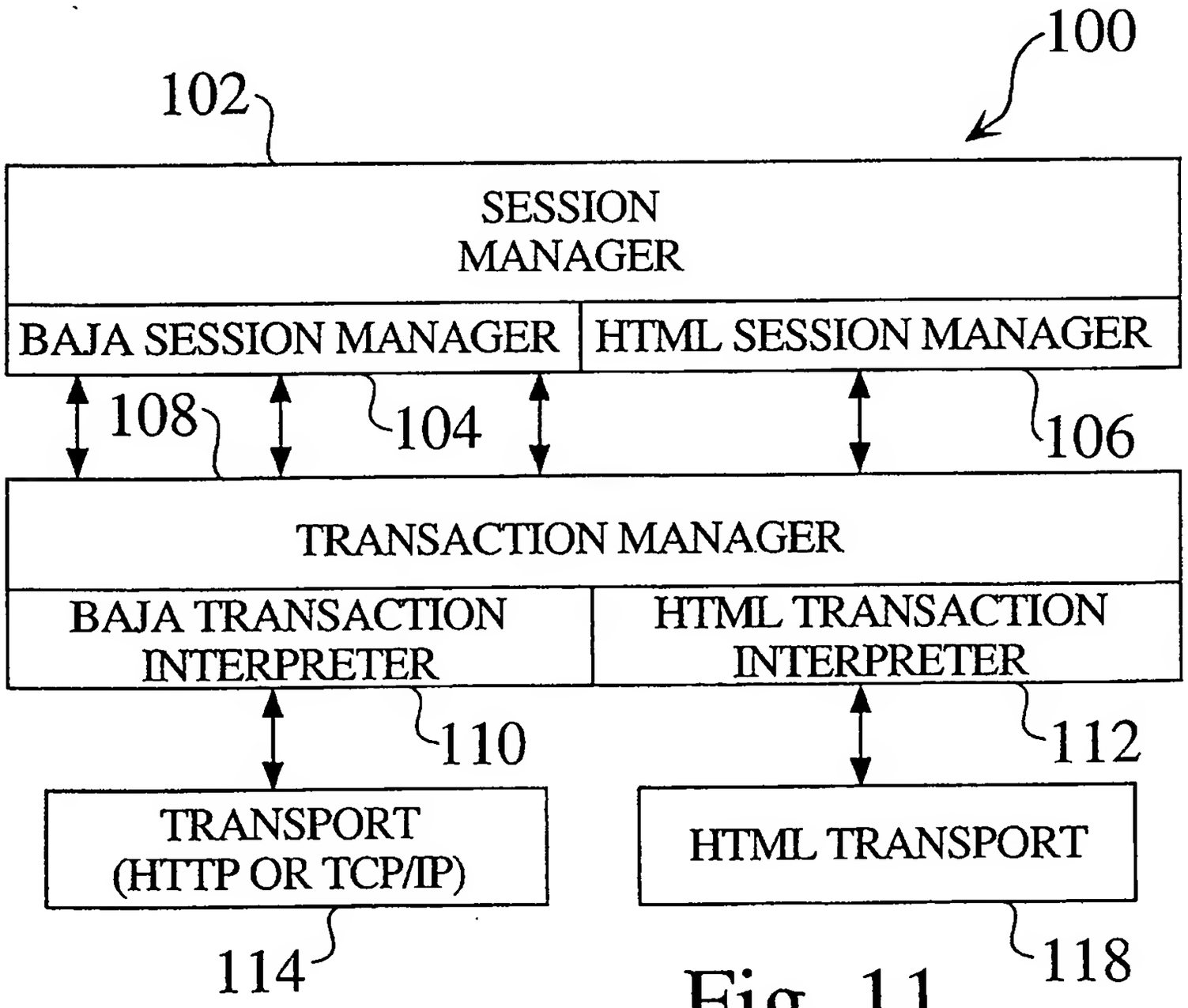
100
102
SESSION MANAGER
BAJA SESSION MANAGER
HTML SESSION MANAGER
108
104
106
TRANSACTION MANAGER
BAJA TRANSACTION INTERPRETER
HTML TRANSACTION INTERPRETER
110
112
TRANSPORT (HTTP OR TCP/IP)
HTML TRANSPORT
114
118

Fig. 11

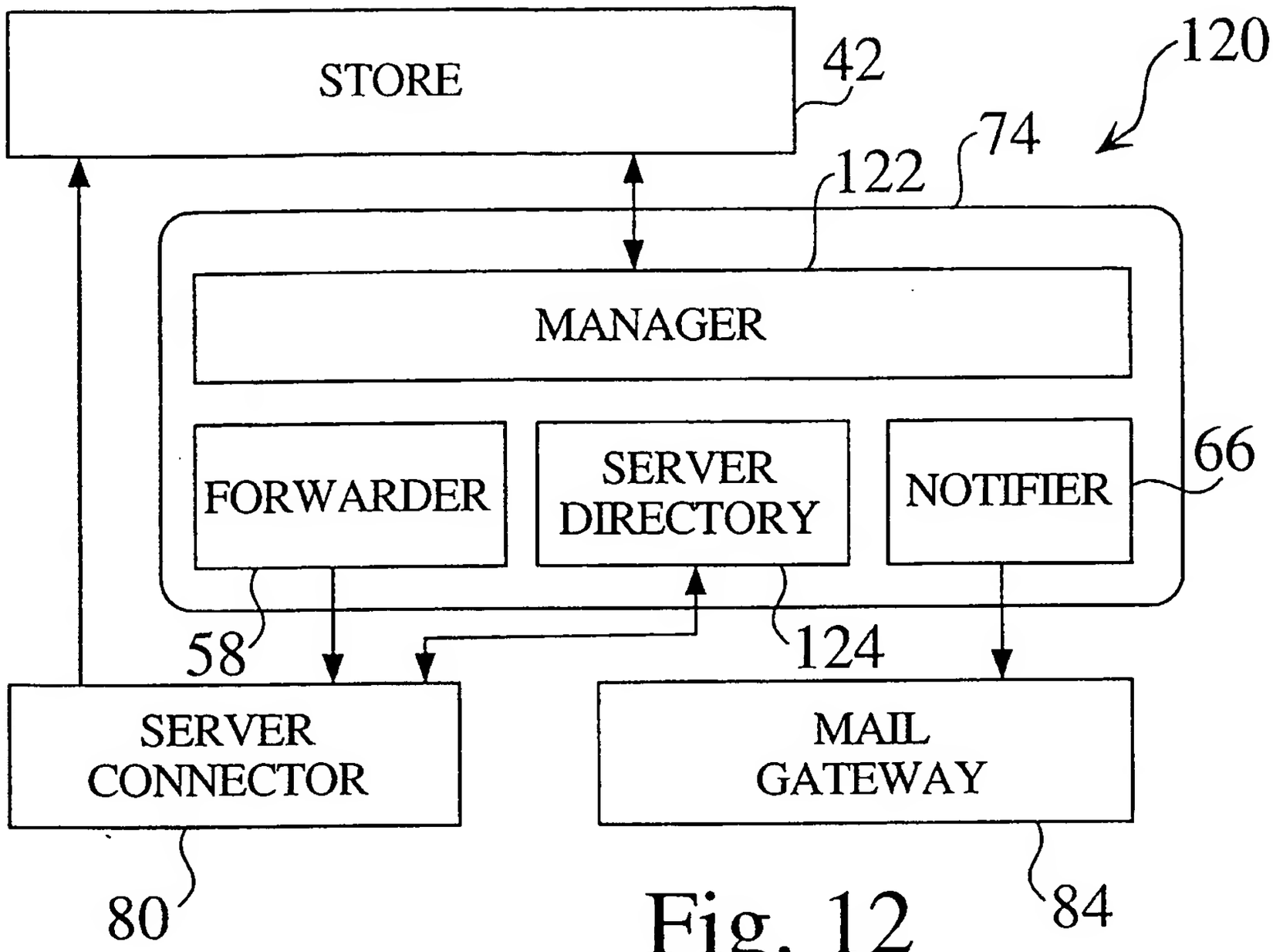
STORE
42
120
74
122
MANAGER
FORWARDER
SERVER DIRECTORY
NOTIFIER
66
58
124
SERVER CONNECTOR
MAIL GATEWAY
80
84

Fig. 12

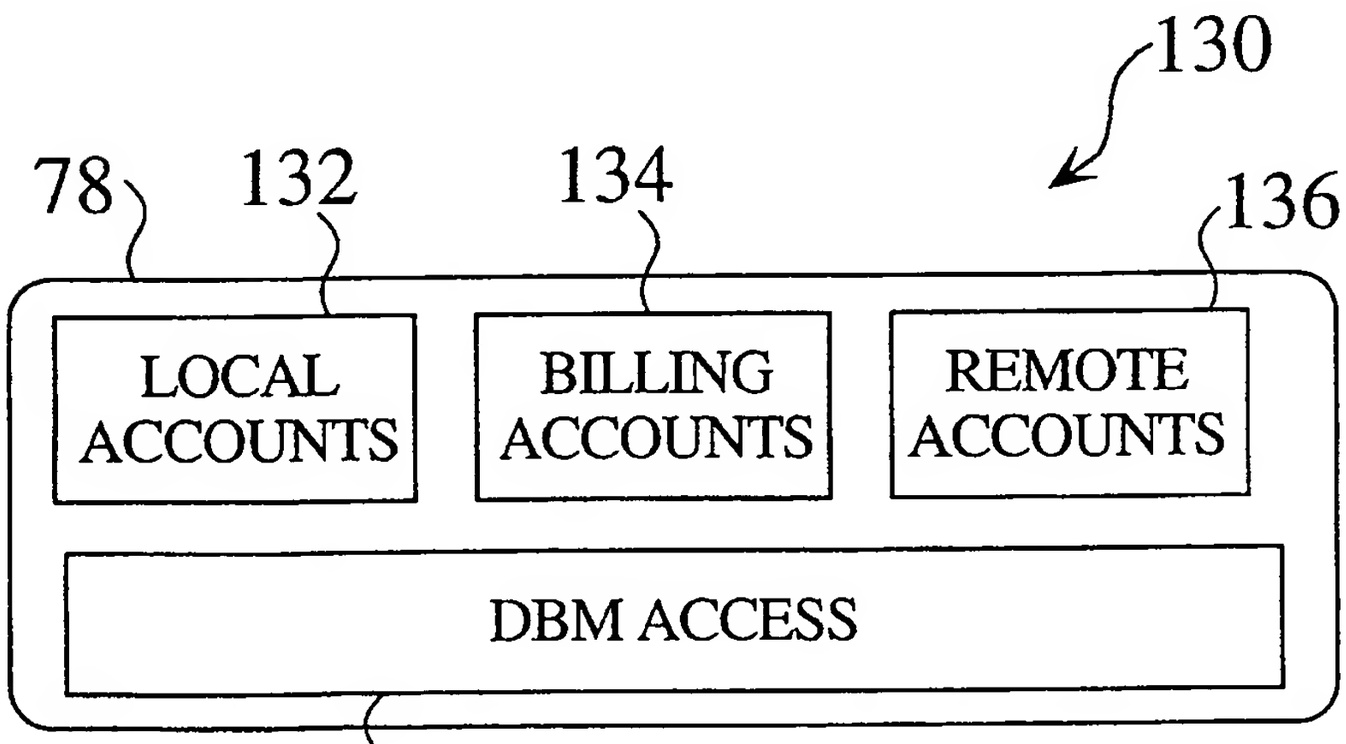
130
78
132
134
136
LOCAL ACCOUNTS
BILLING ACCOUNTS
REMOTE ACCOUNTS
DBM ACCESS
138

Fig. 13

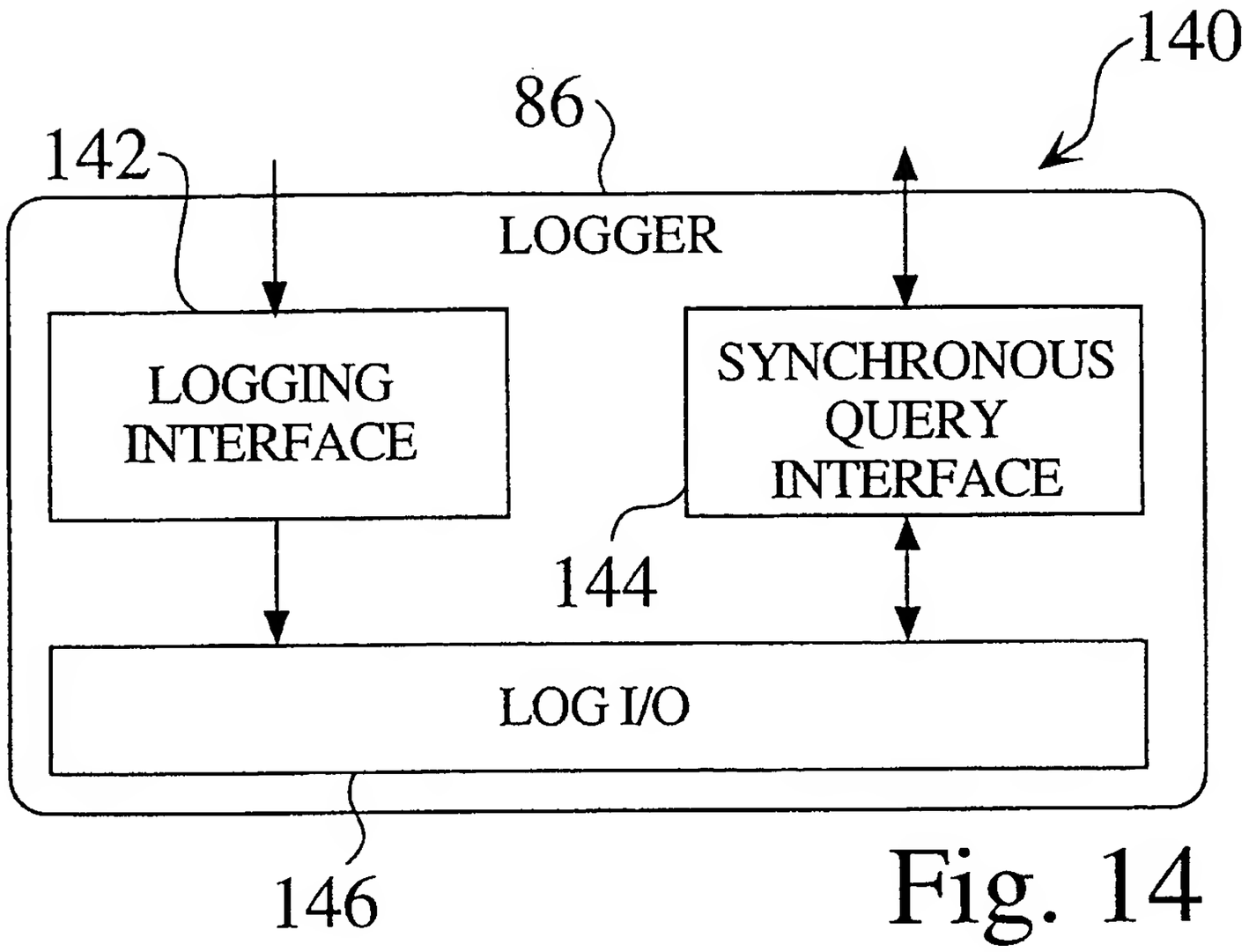
140
86
142
LOGGER
LOGGING INTERFACE
SYNCHRONOUS QUERY INTERFACE
144
LOG I/O
146

Fig. 14

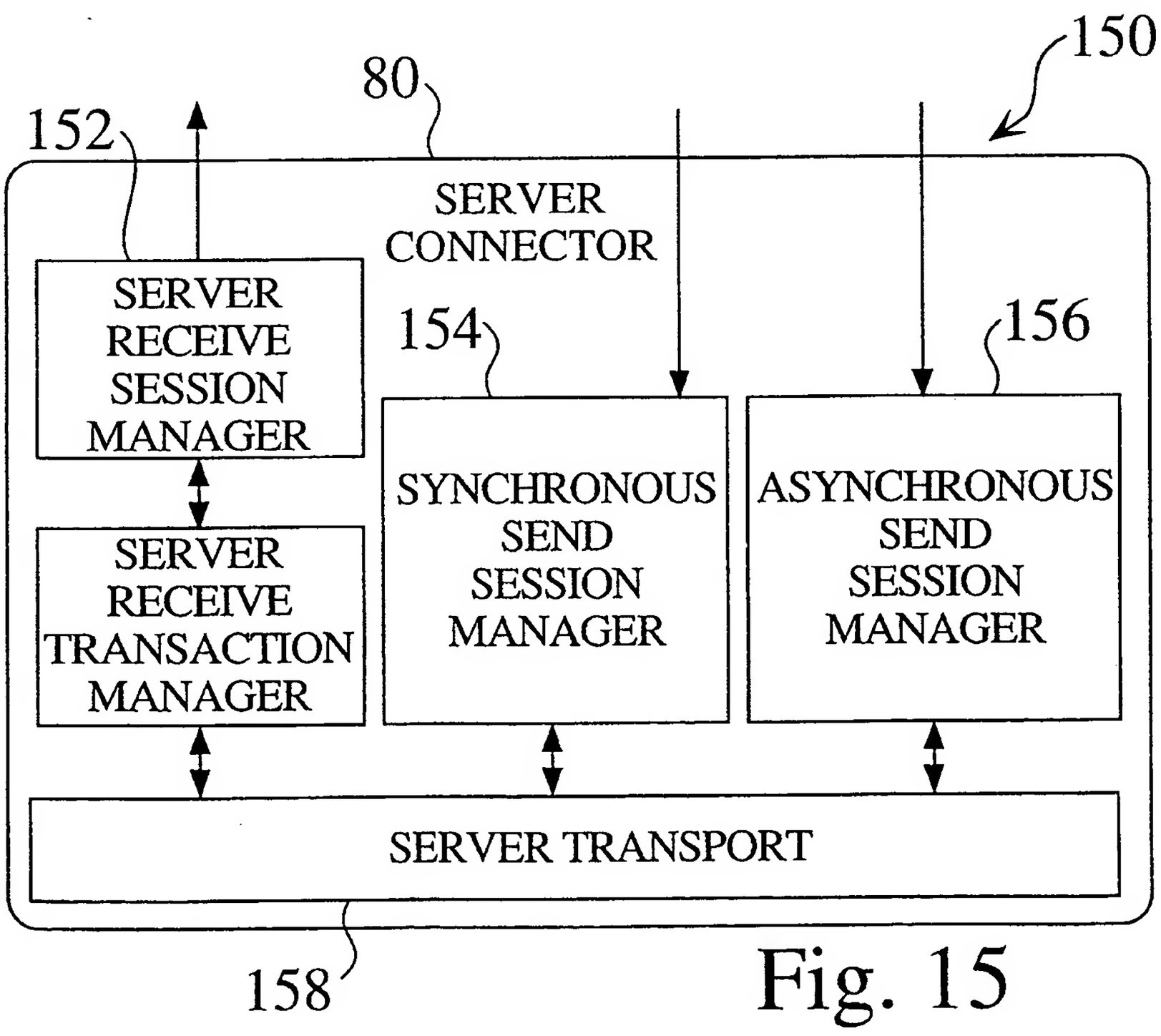
150
80
152
SERVER CONNECTOR
SERVER RECEIVE SESSION MANAGER
154
156
SERVER RECEIVE TRANSACTION MANAGER
SYNCHRONOUS SEND SESSION MANAGER
ASYNCHRONOUS SEND SESSION MANAGER
SERVER TRANSPORT
158

Fig. 15

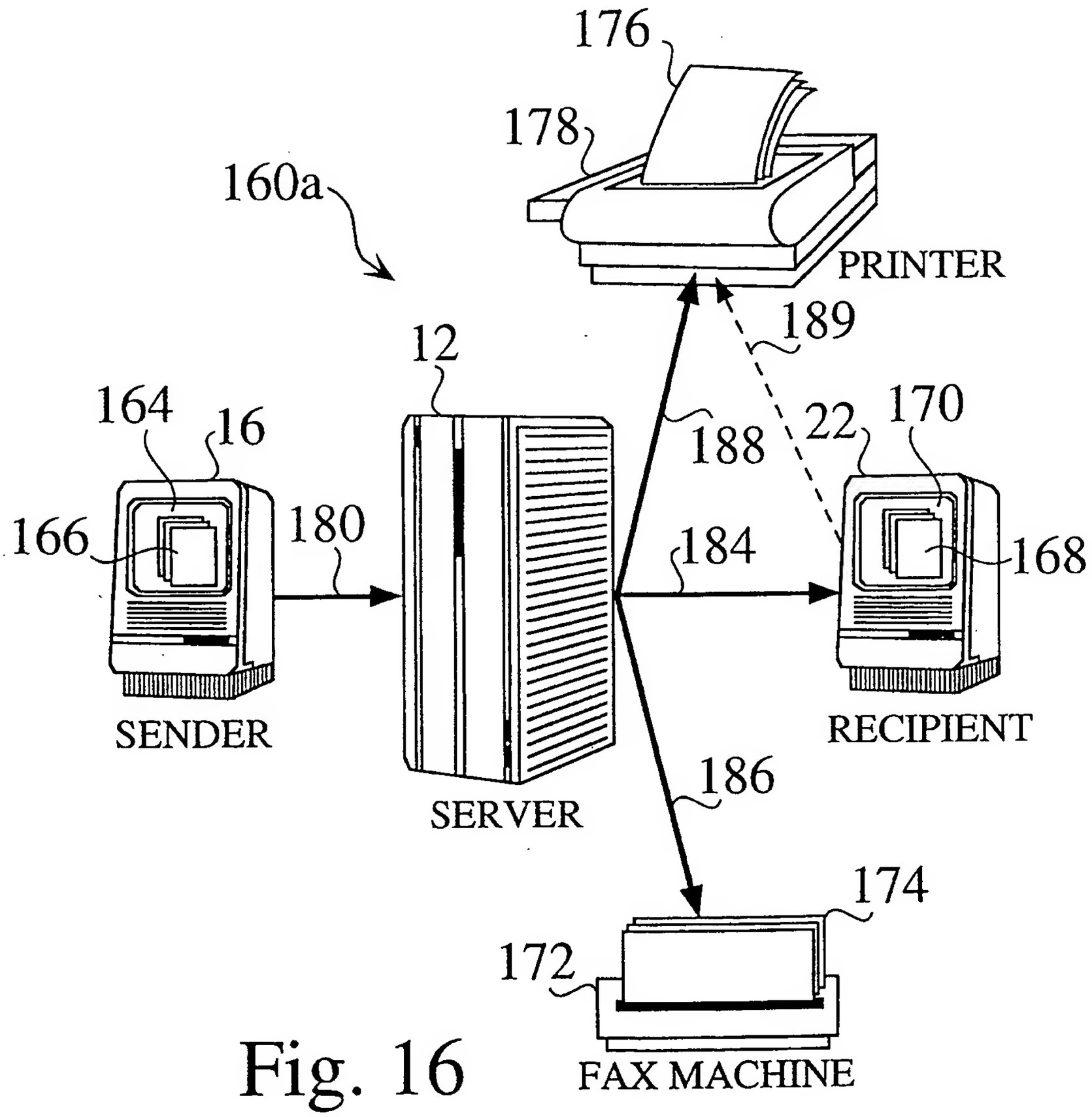
176
178
160a
PRINTER
189
12
164
16
188
22
170
166
180
184
168
SENDER
SERVER
RECIPIENT
186
174
172
FAX MACHINE

Fig. 16

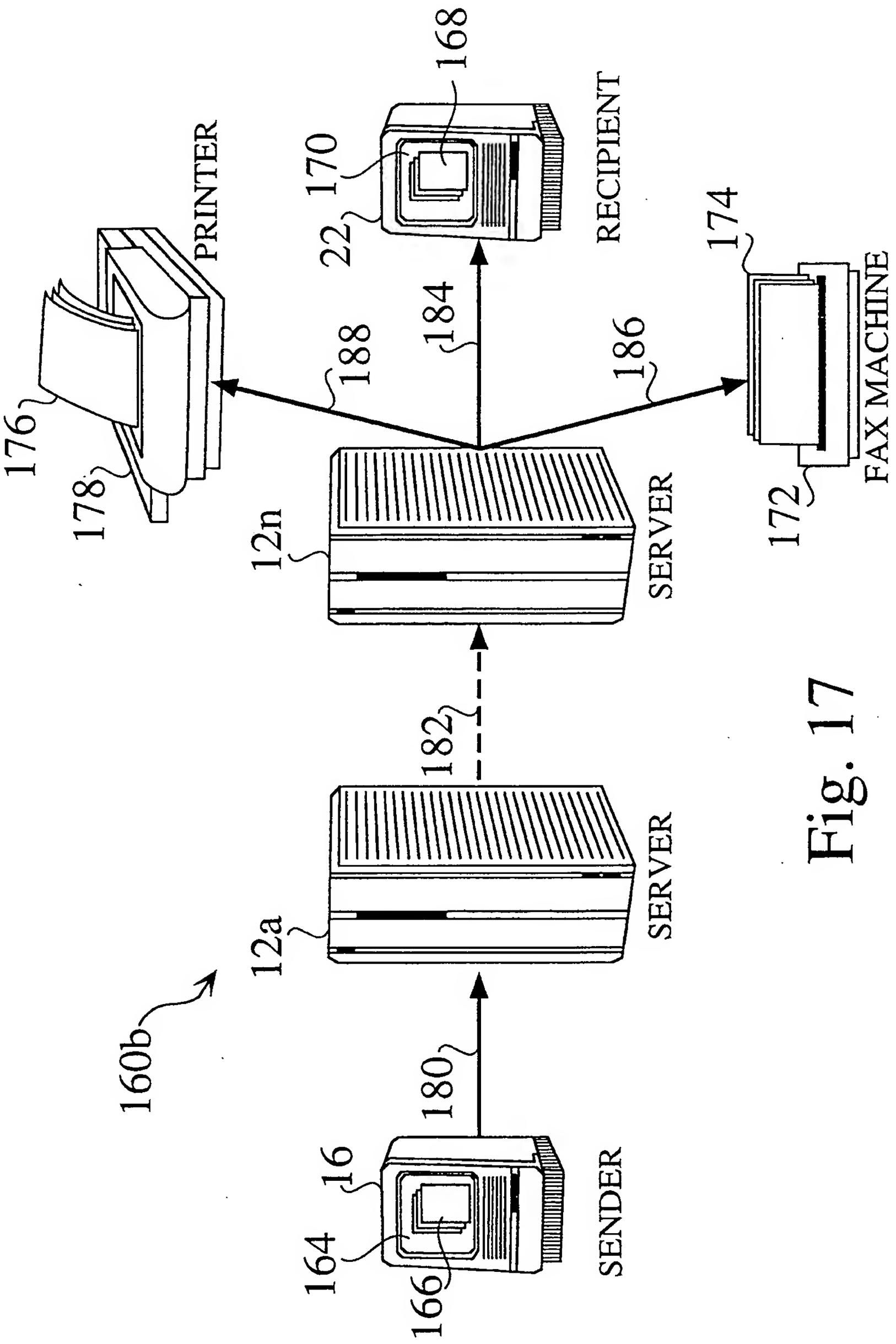
PRINTER
168
170
22
RECIPIENT
174
184
188
186
FAX MACHINE
176
178
172
12n
SERVER
182
12a
SERVER
160b
180
16
164
166
SENDER

Fig. 17

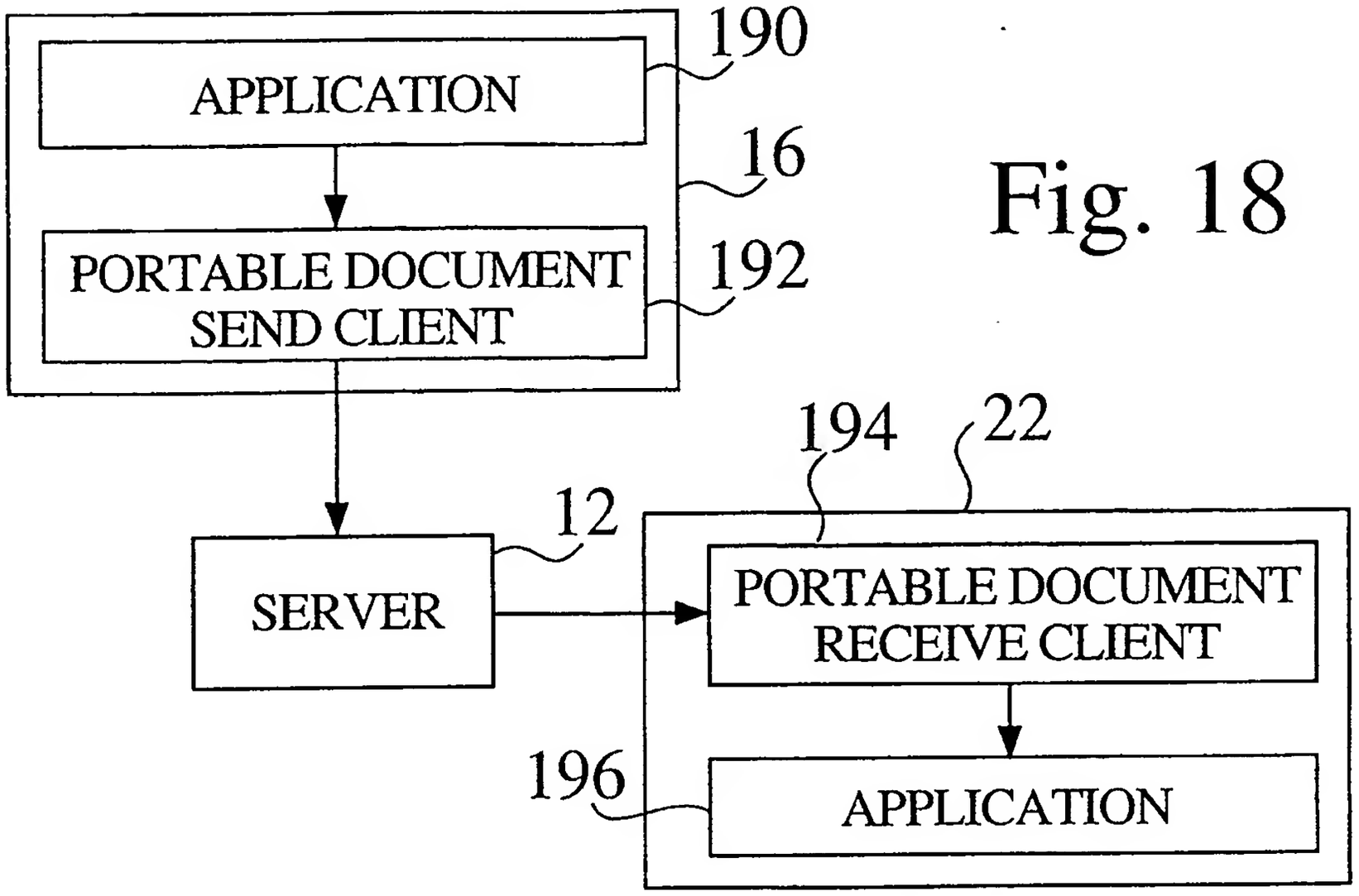
APPLICATION
190
16
PORTABLE DOCUMENT SEND CLIENT
192
194
22
12
SERVER
PORTABLE DOCUMENT RECEIVE CLIENT
196
APPLICATION

Fig. 18

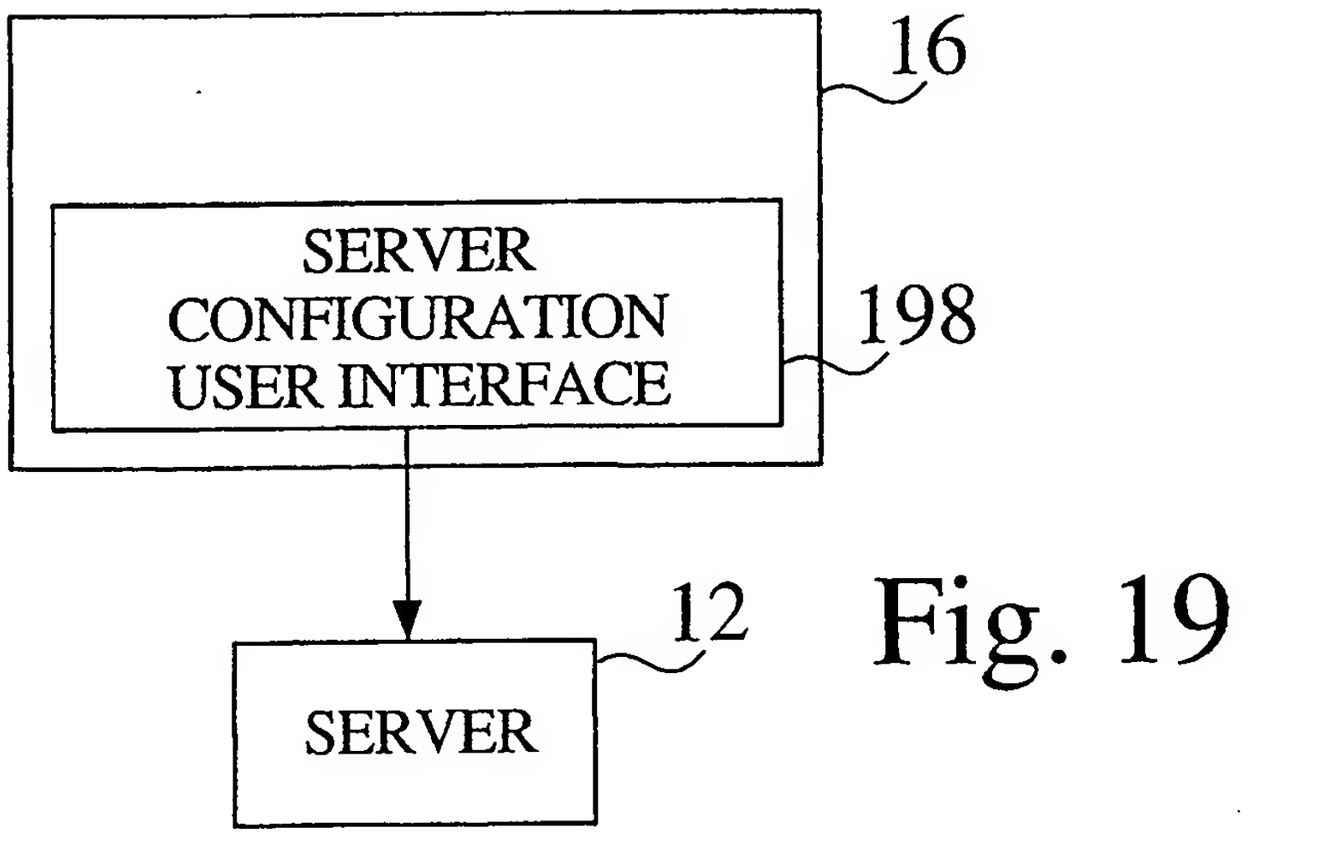
16
SERVER CONFIGURATION USER INTERFACE
198
12
SERVER

Fig. 19

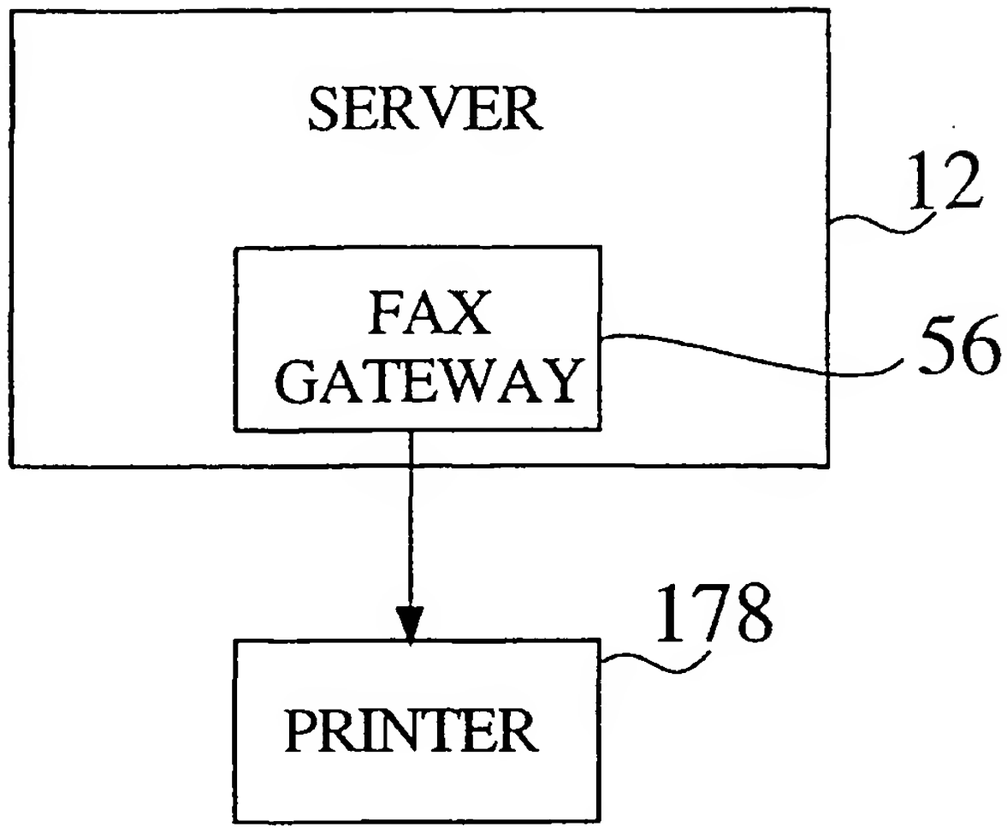
SERVER
12
FAX
GATEWAY
56
178
PRINTER

Fig. 20

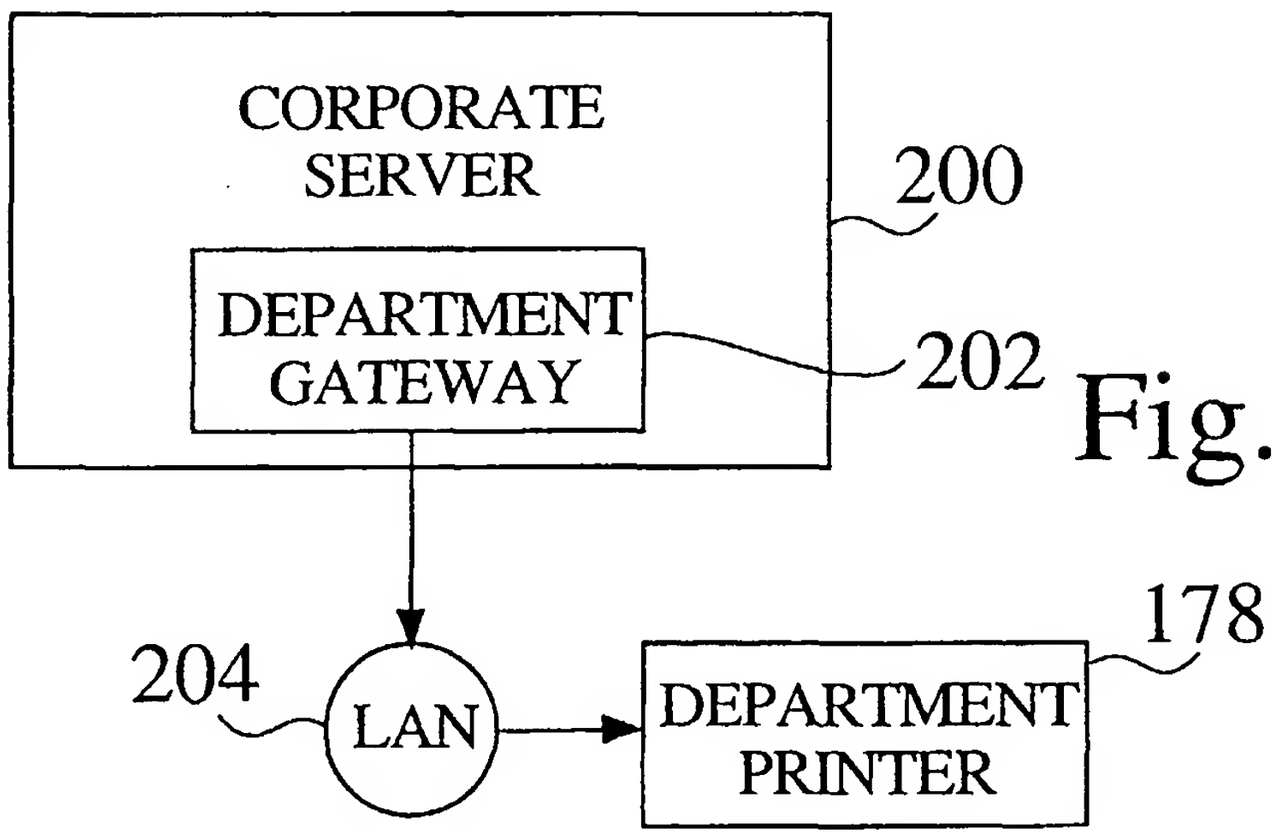
CORPORATE
SERVER
200
DEPARTMENT
GATEWAY
202
204
LAN
178
DEPARTMENT
PRINTER

Fig. 21

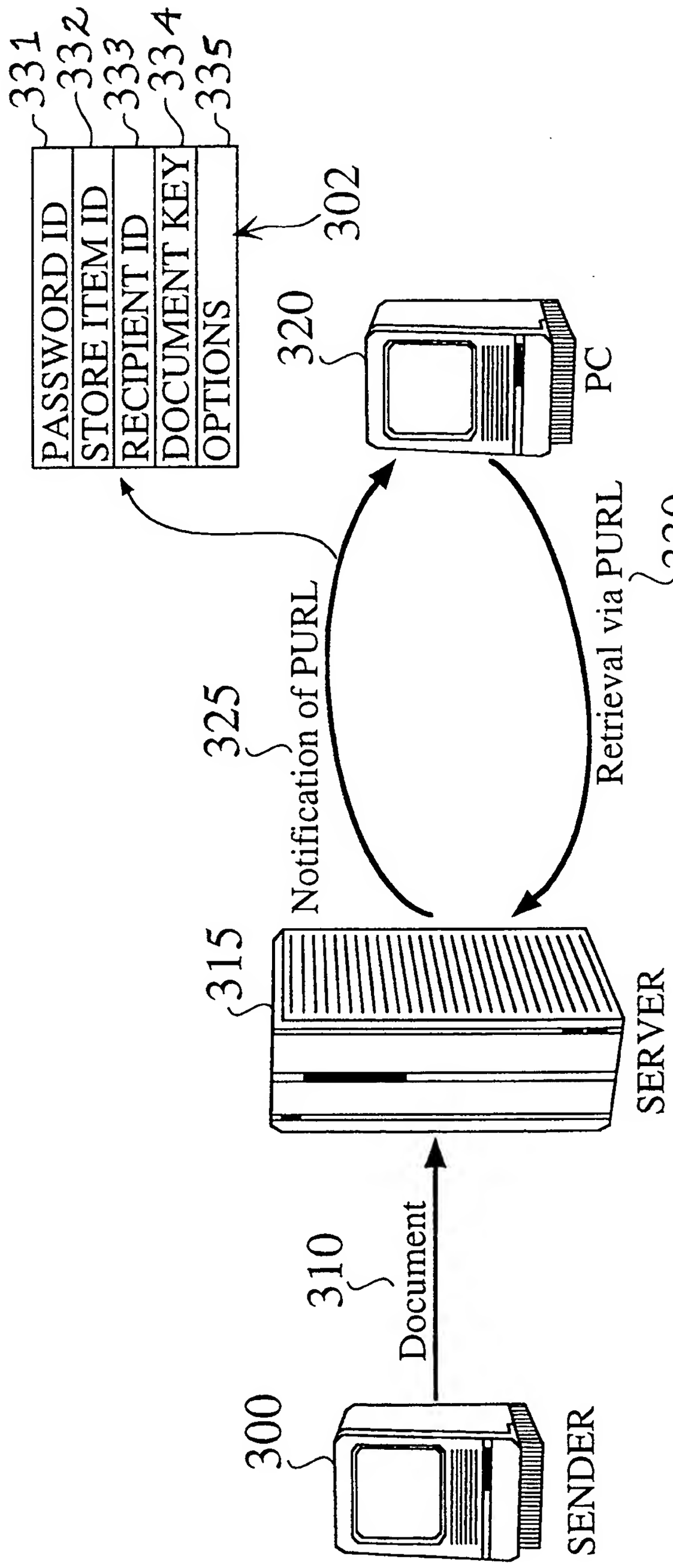
331
332
333
334
335
PASSWORD ID
STORE ITEM ID
RECIPIENT ID
DOCUMENT KEY
OPTIONS
302
320
PC
Retrieval via PURL
330
325
Notification of PURL
315
SERVER
310
Document
300
SENDER

Fig. 22

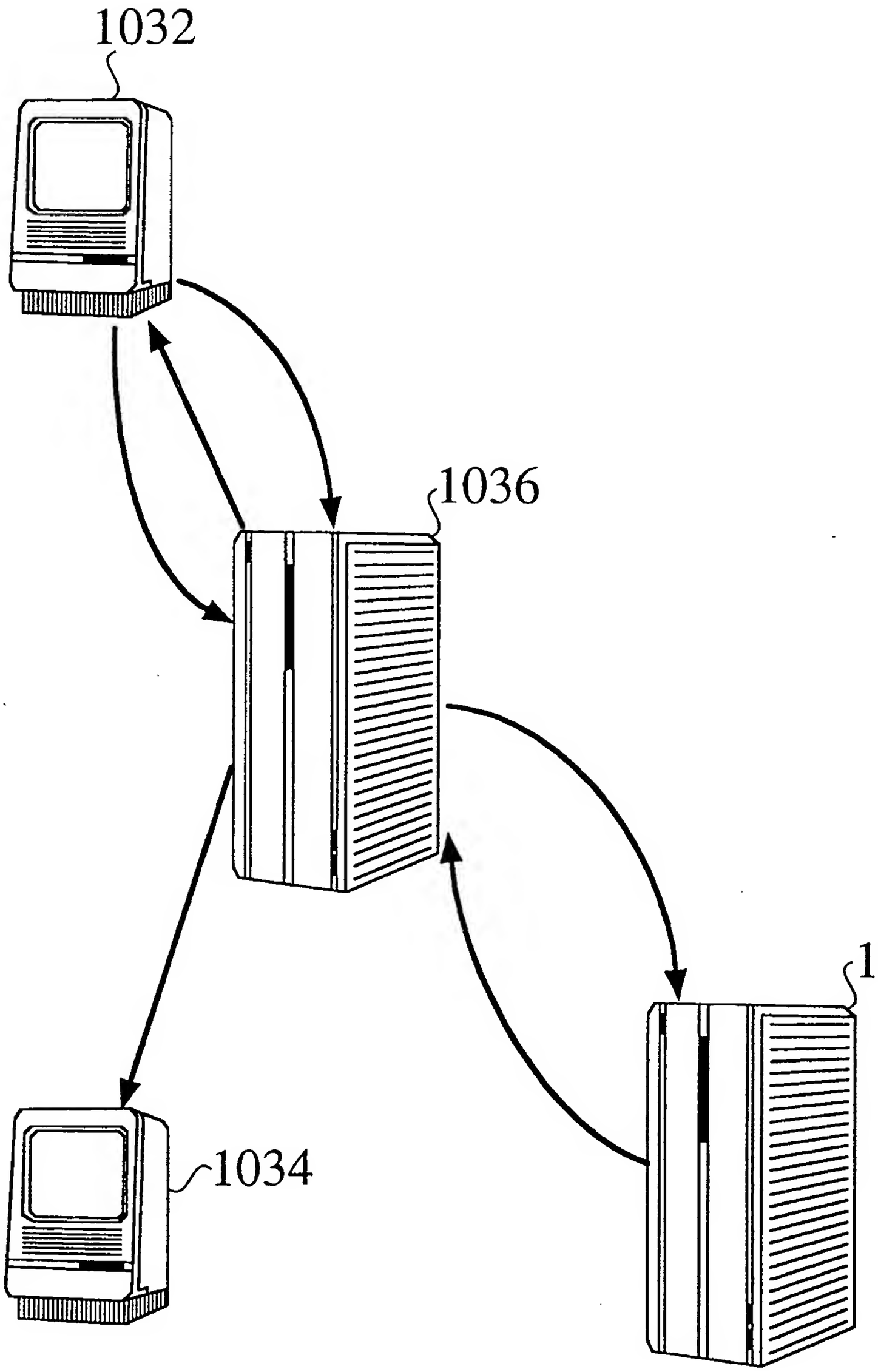

Fig. 23

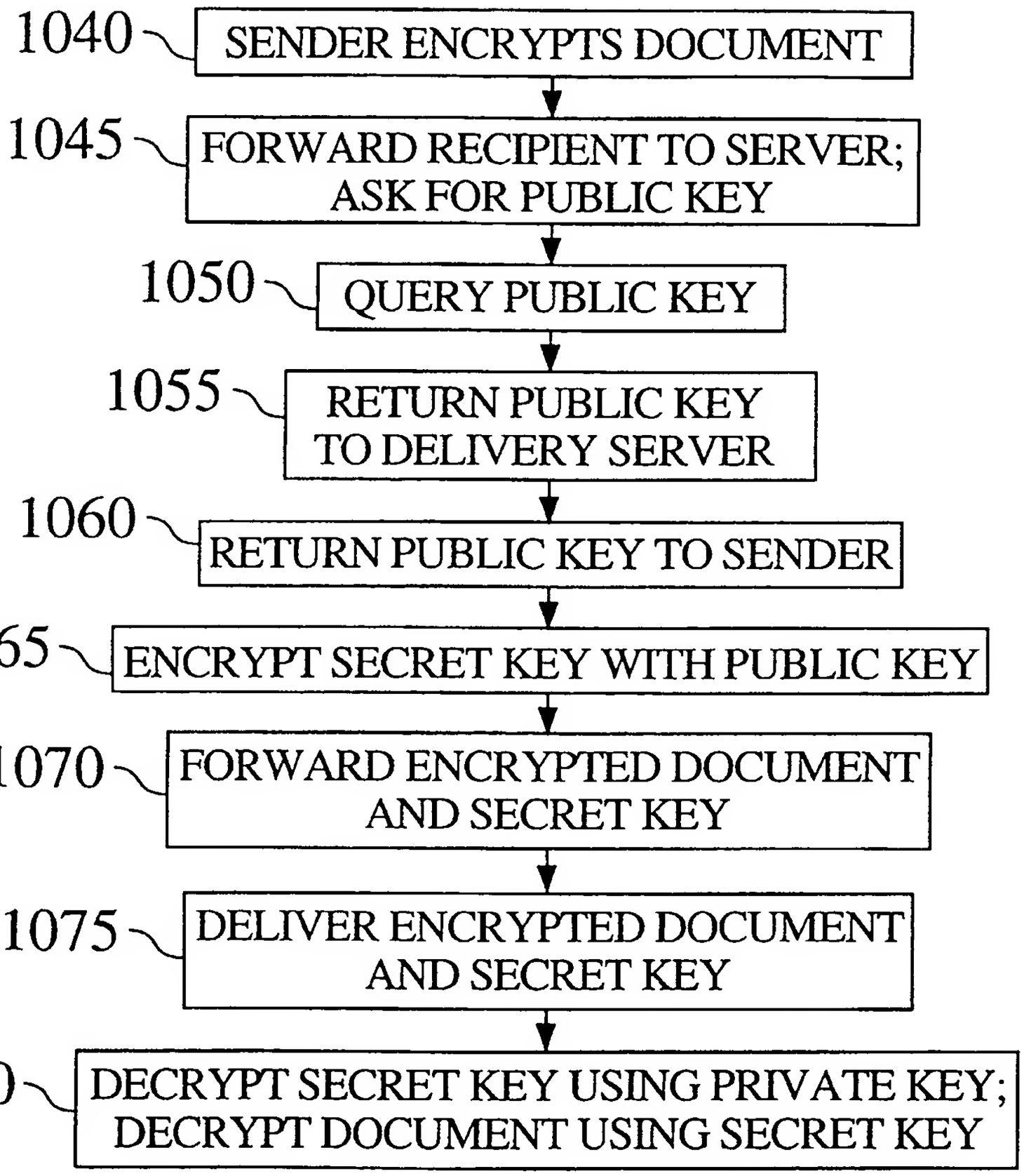
1040
SENDER ENCRYPTS DOCUMENT
1045
FORWARD RECIPIENT TO SERVER; ASK FOR PUBLIC KEY
1050
QUERY PUBLIC KEY
1055
RETURN PUBLIC KEY TO DELIVERY SERVER
1060
RETURN PUBLIC KEY TO SENDER
1065
ENCRYPT SECRET KEY WITH PUBLIC KEY
1070
FORWARD ENCRYPTED DOCUMENT AND SECRET KEY
1075
DELIVER ENCRYPTED DOCUMENT AND SECRET KEY
1080
DECRYPT SECRET KEY USING PRIVATE KEY; DECRYPT DOCUMENT USING SECRET KEY

Fig. 24

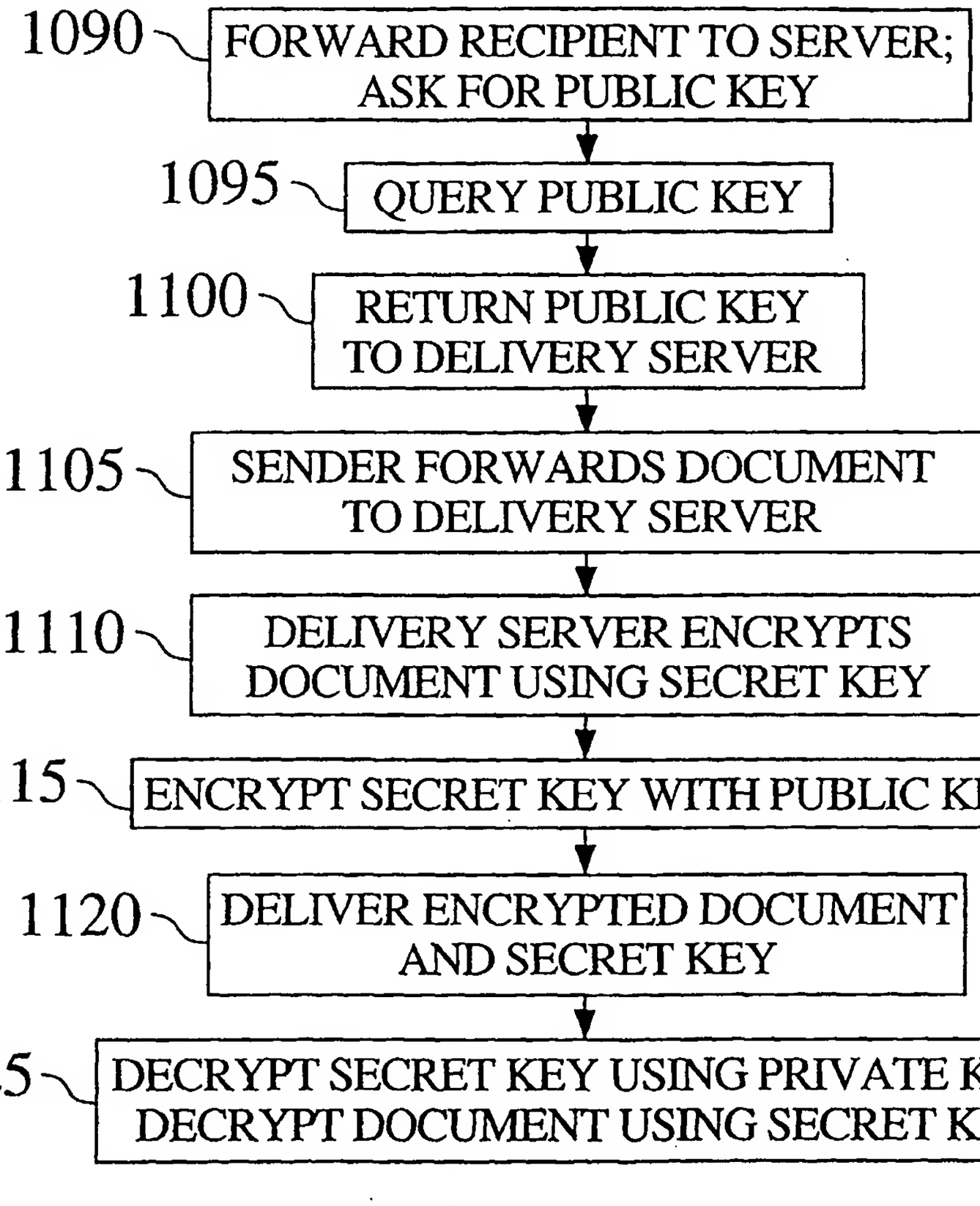
1090
FORWARD RECIPIENT TO SERVER; ASK FOR PUBLIC KEY
1095
QUERY PUBLIC KEY
1100
RETURN PUBLIC KEY TO DELIVERY SERVER
1105
SENDER FORWARDS DOCUMENT TO DELIVERY SERVER
1110
DELIVERY SERVER ENCRYPTS DOCUMENT USING SECRET KEY
1115
ENCRYPT SECRET KEY WITH PUBLIC KEY
1120
DELIVER ENCRYPTED DOCUMENT AND SECRET KEY
1125
DECRYPT SECRET KEY USING PRIVATE KEY; DECRYPT DOCUMENT USING SECRET KEY

Fig. 25